

# A “little” tour of assembly methods

Antoine Limasset CRISTAL, Université de Lille, CNRS,  
France

(ii)

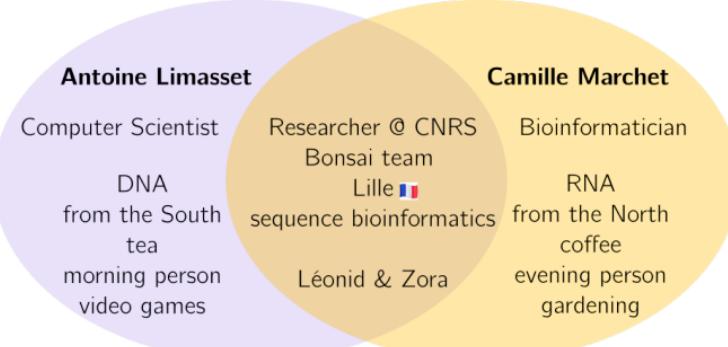
antoine.limasset@gmail.com  
camille.marchet@univ-lille.fr



@npmalfoy  
@camillemrcht

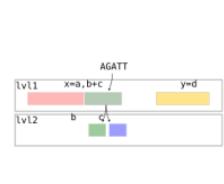
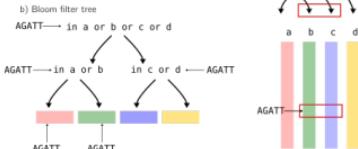
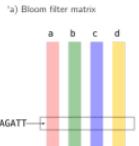
(iii)





Things you might want to discuss with us:

- methodological/scalability questions?
- young kids/work balance
- impostor syndrome when doing CS stuff
- nice tenured positions in France



The screenshot shows a "Sequence Search" interface. On the left, there is a sidebar titled "Select one or more indices" with several options listed. The main area has a search bar with the placeholder "Your request" and a "Search" button below it.

## • Content of this course

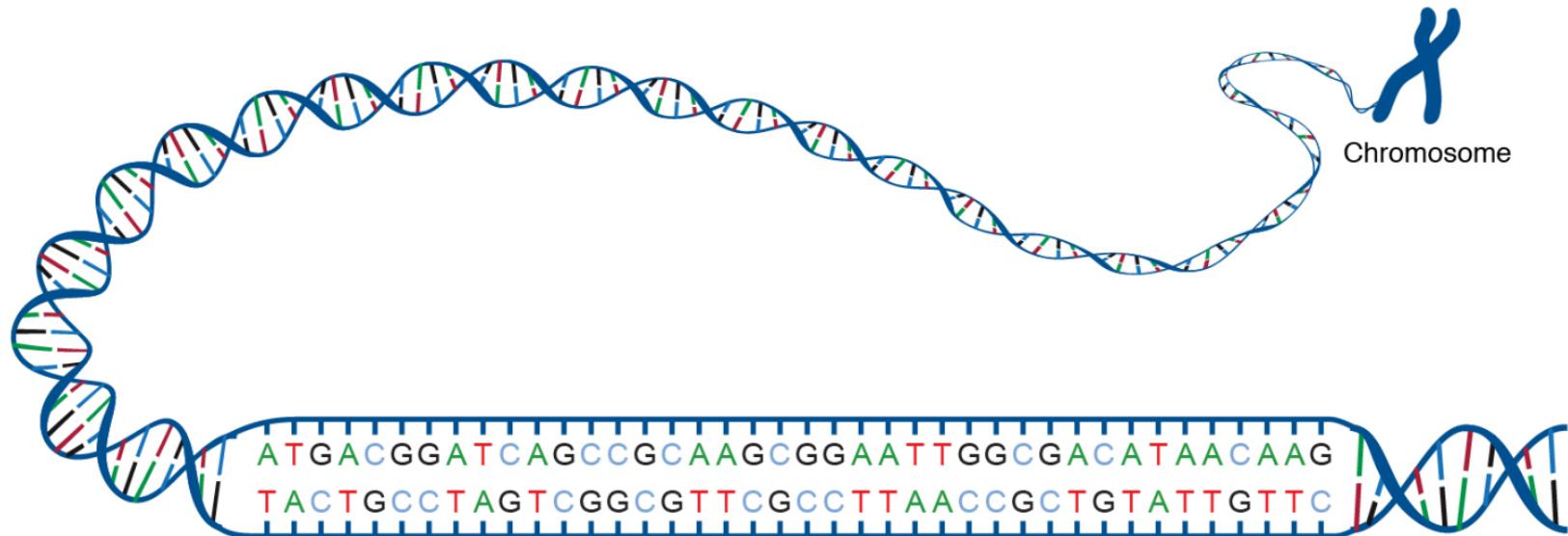
- How to reconstruct a genome with sequencing data?
- What are the main challenges?
- Which solutions have been proposed?

Bingo: find a book that we both love (French title).



genome size:  $\sim 40$  gigabases

- Accessing a genome



- Why do we need assembly?



**Laura Landweber** @LandweberLab · Jan 2



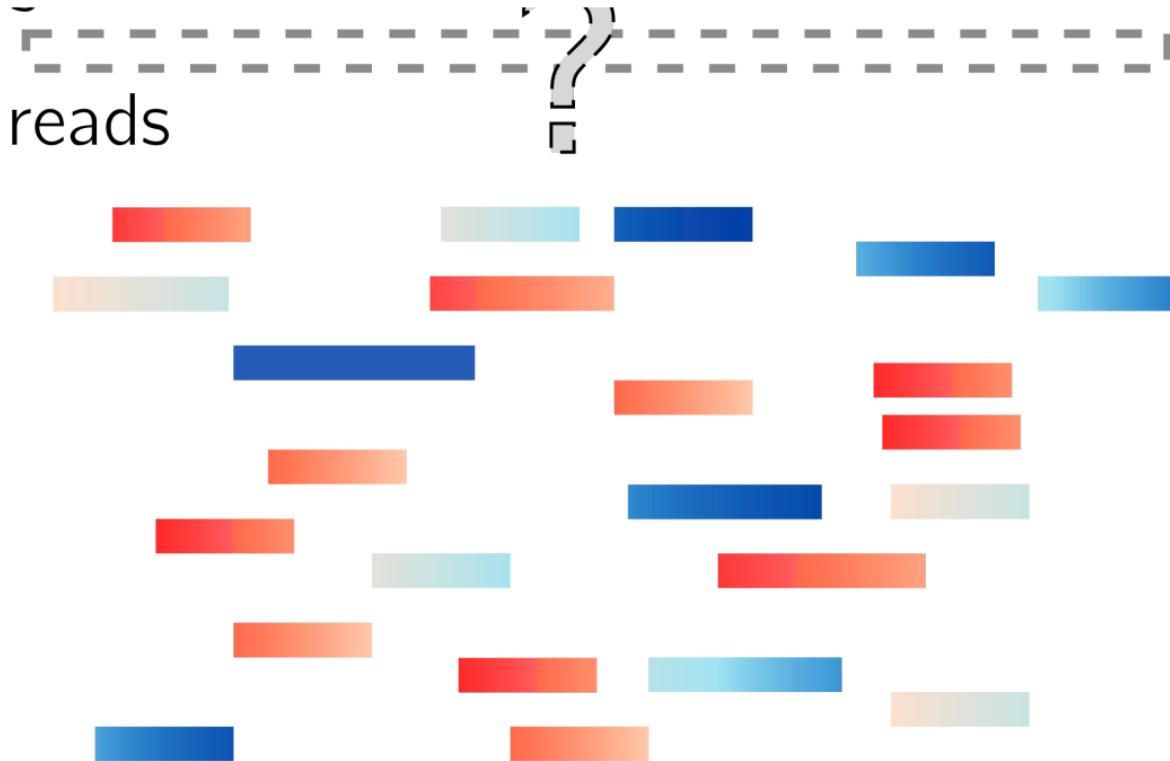
Our newest version of *Oxytricha*'s somatic genome is out ([rdcu.be/bZNfC](https://rdcu.be/bZNfC)) and has 18,617 distinct chromosomes. That's 2000 more than we previously published in [doi.org/10.1371/journal....](https://doi.org/10.1371/journal.pbio.3000001) PacBio captured most chromosomes in single reads: Genome sequence, No assembly required

- Reads are subsequences from the genome

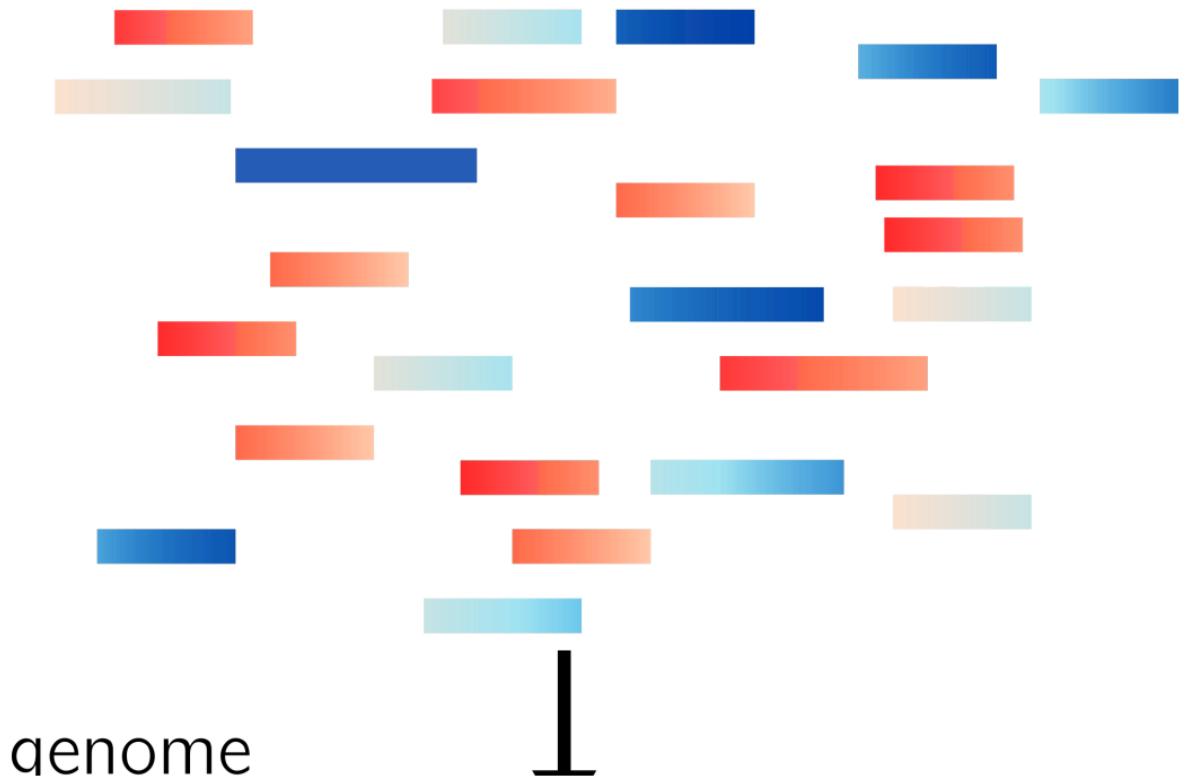
genome



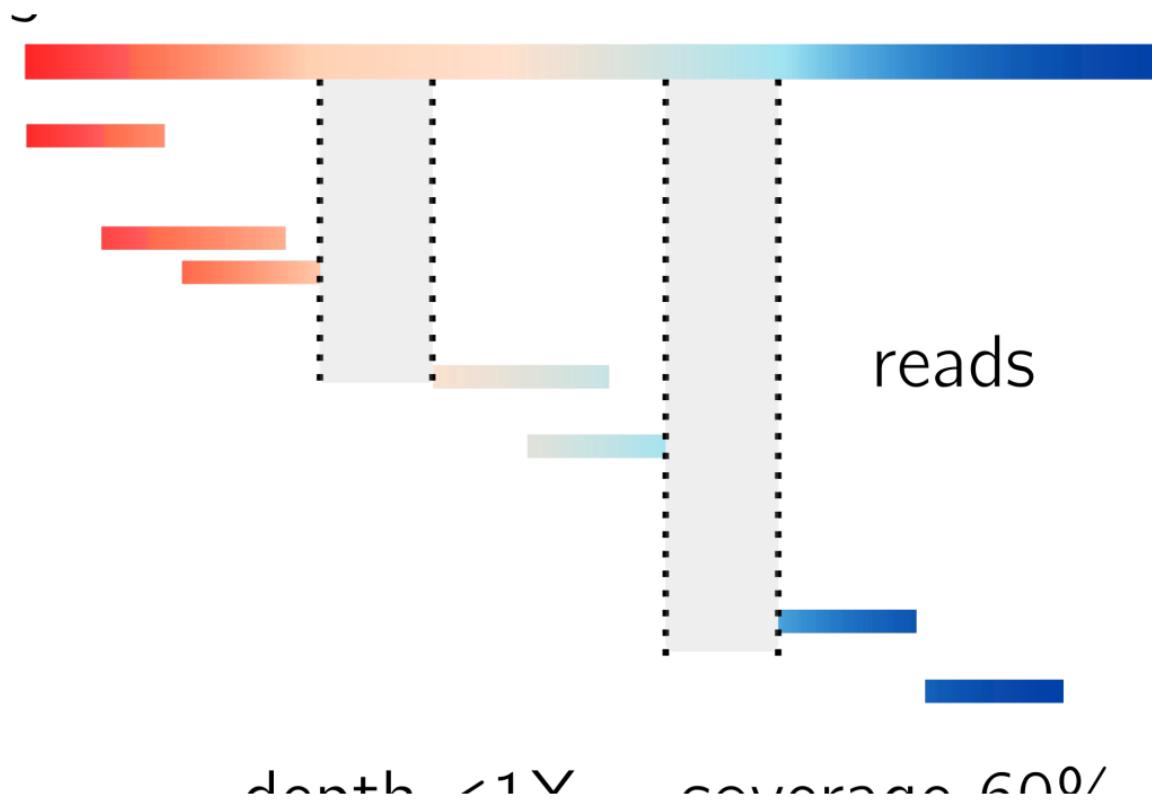
- Reads are shuffled subsequences from the genome



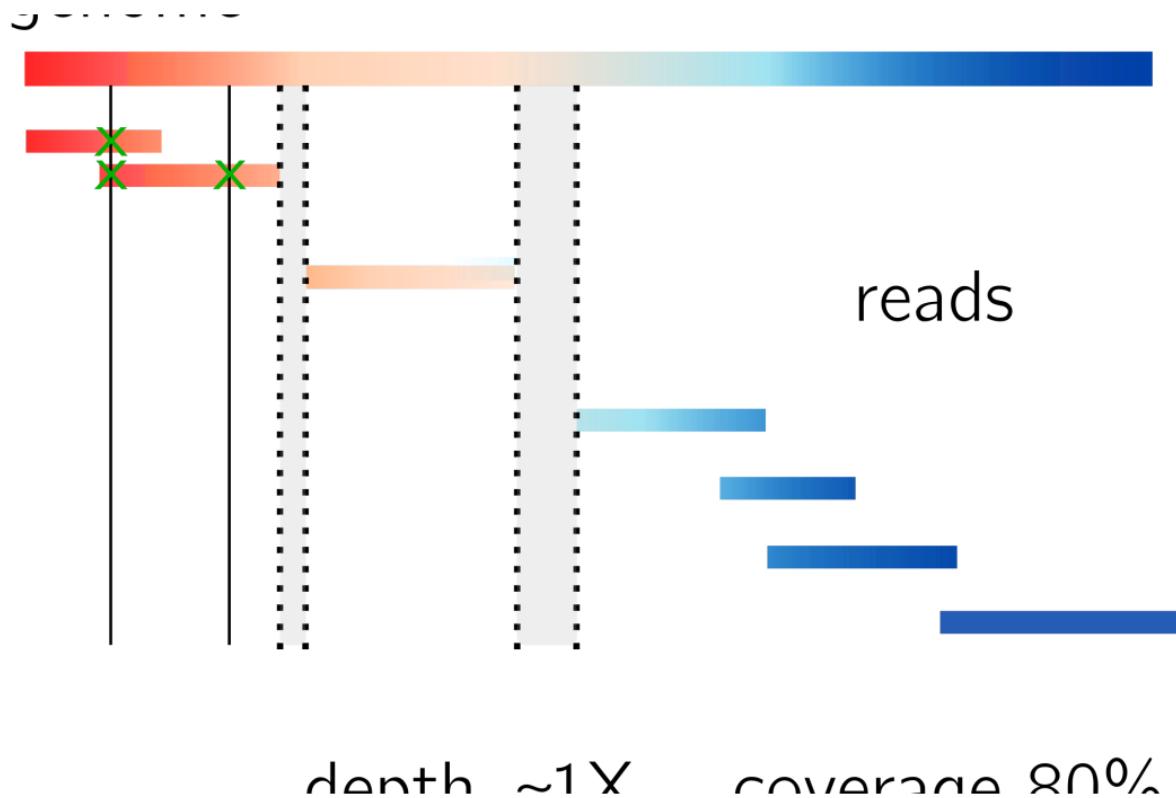
- Genome assembly task



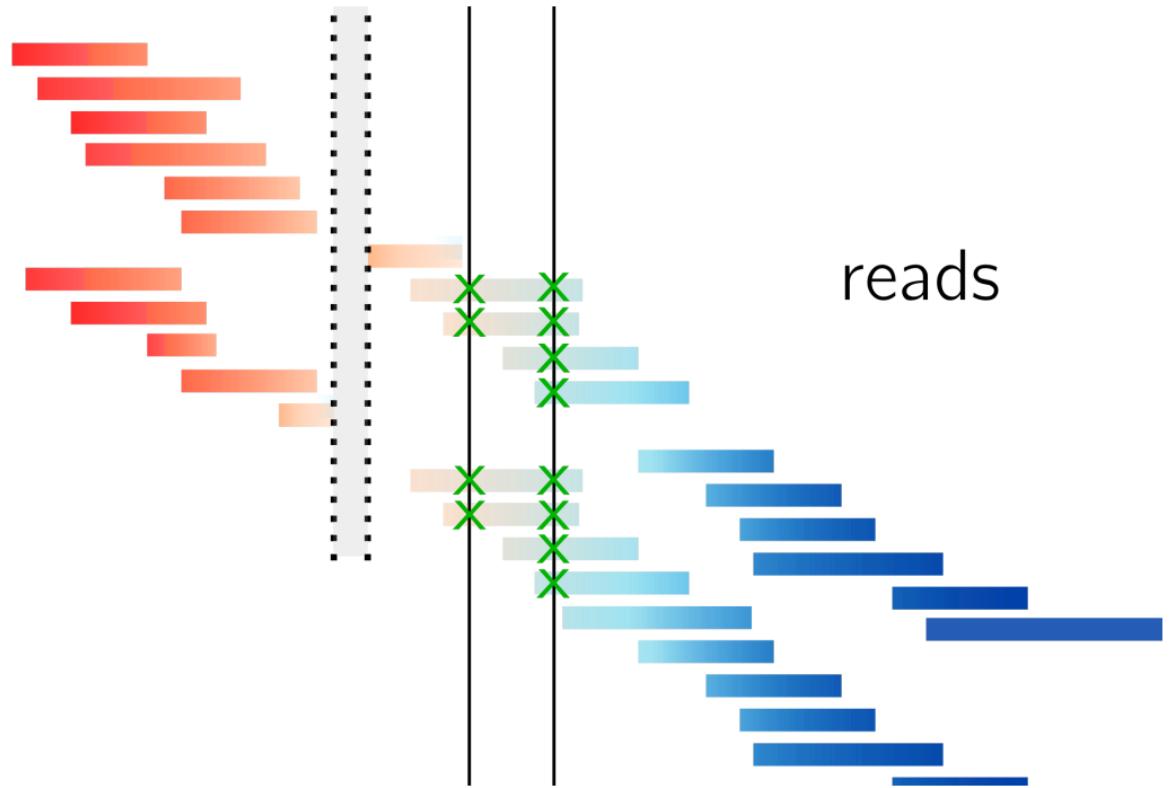
- Genome sequencing: depth & coverage



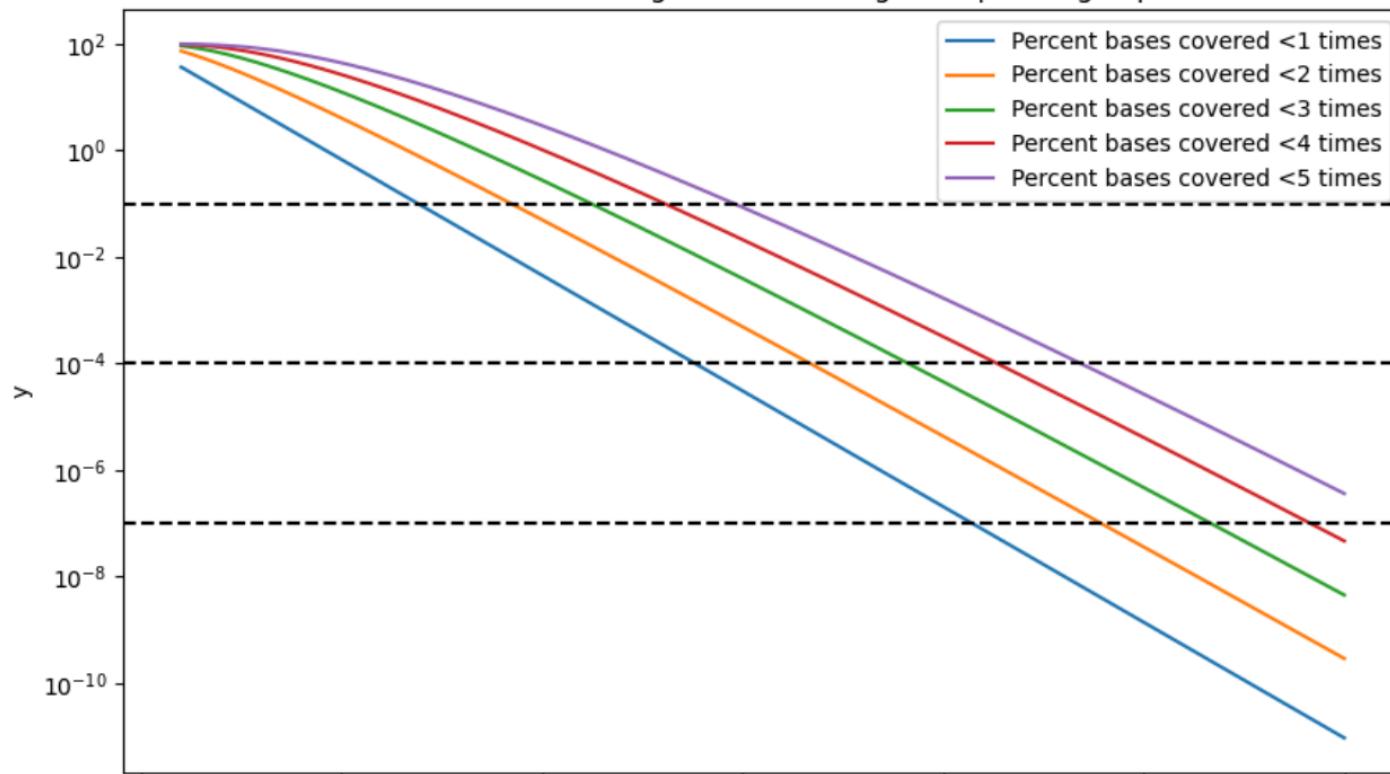
- Genome sequencing: depth & coverage



- Genome sequencing: depth & coverage



- Poisson law



- **Poisson law (ii)**

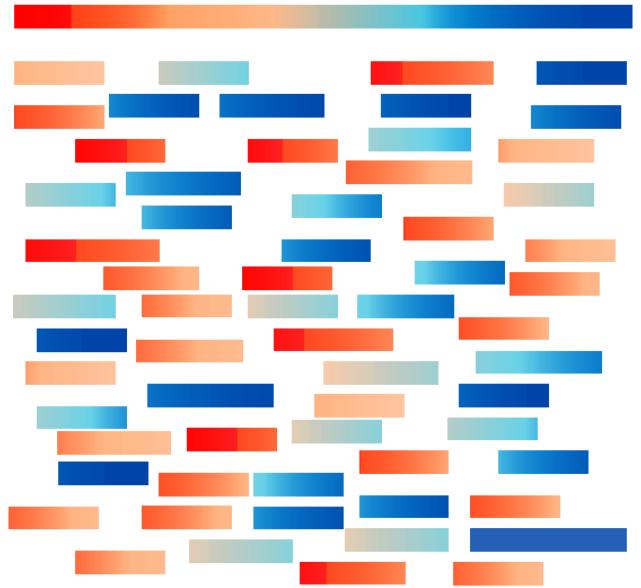
30X are often required for assembly projects

- First experiment: *Long, perfect boy's genome*



Genome size  
1 billion bases

(only for the record, we actually don't have it)

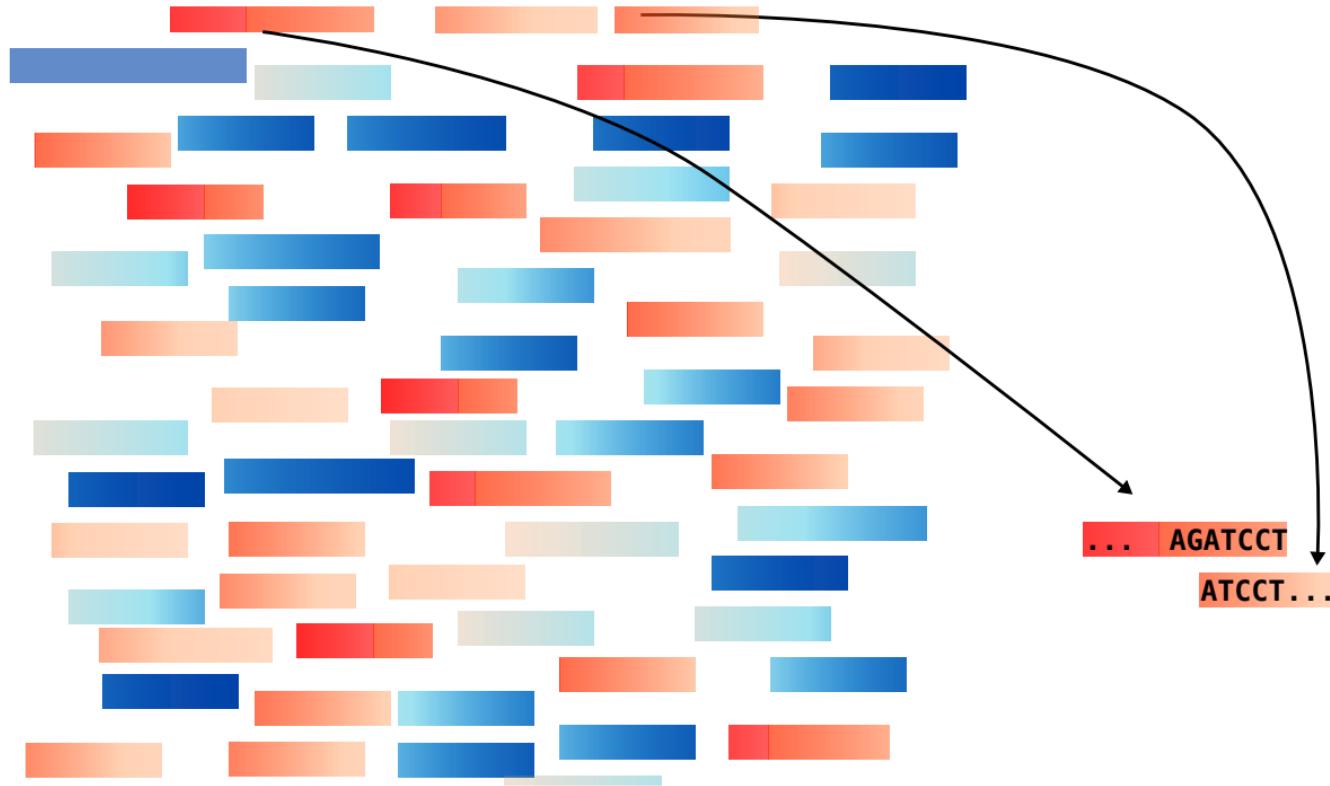


Reads  
10 million

- Order according to overlaps

Overlapping reads are likely successive part of the genome

- Order according to overlaps (ii)



- How to compute the overlaps? Alignment?

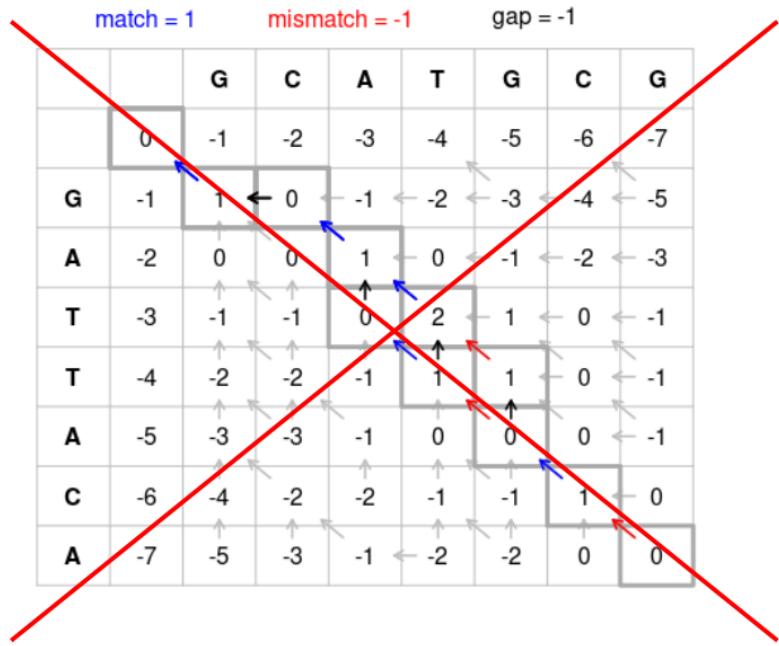
**GATTACA**  
 ↓  
**GCATGCG**

compute  
overlap?

		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1 ← 0	-1	-2	-3	-4	-5	-6
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

match = 1      mismatch = -1      gap = -1

- How to compute the overlaps? Quick exact match!



GATTACA

↑  
↓  
GCATGCG  
↑  
↓

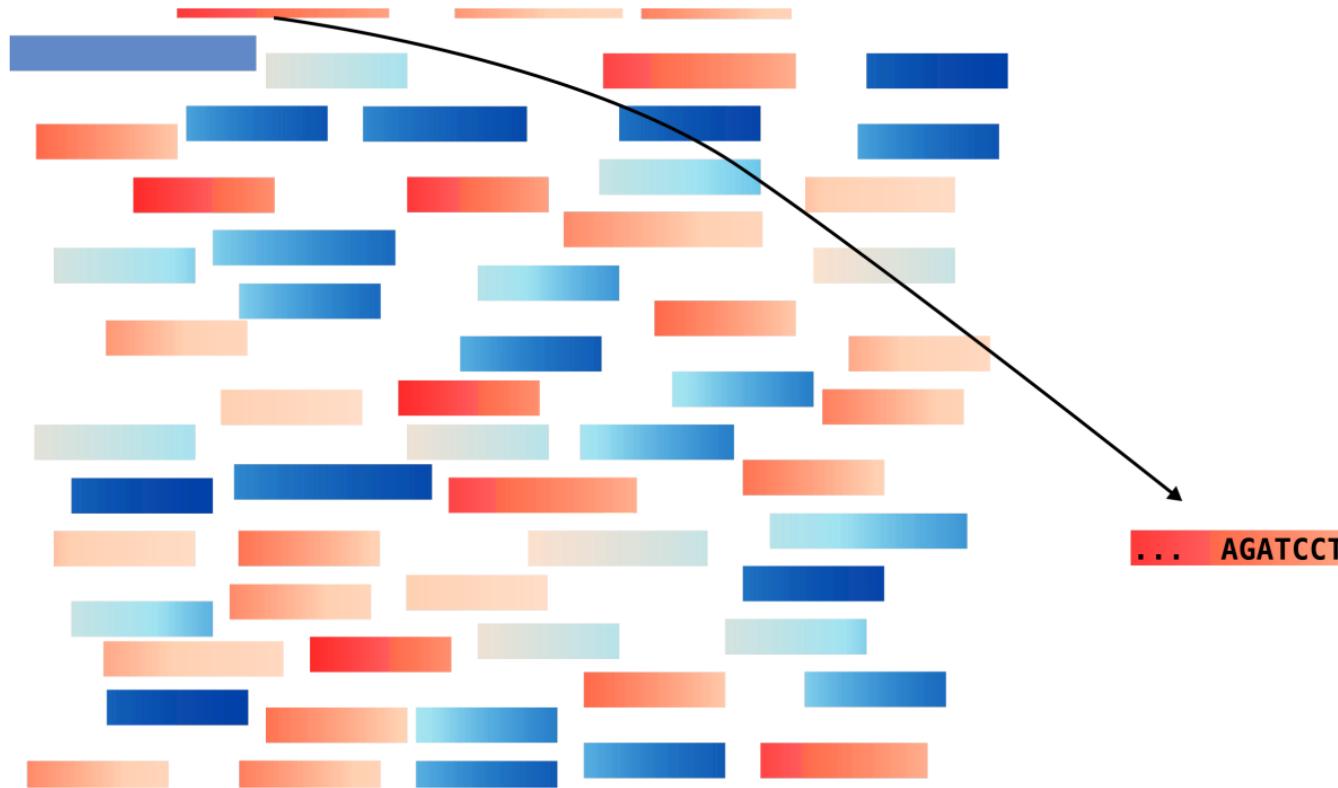
GATTACA

↑  
↓  
↑  
↓  
↑  
↓  
↑  
↓  
↑  
↓  
GATTACA

- Check all reads for overlaps

For a given read, scan all others

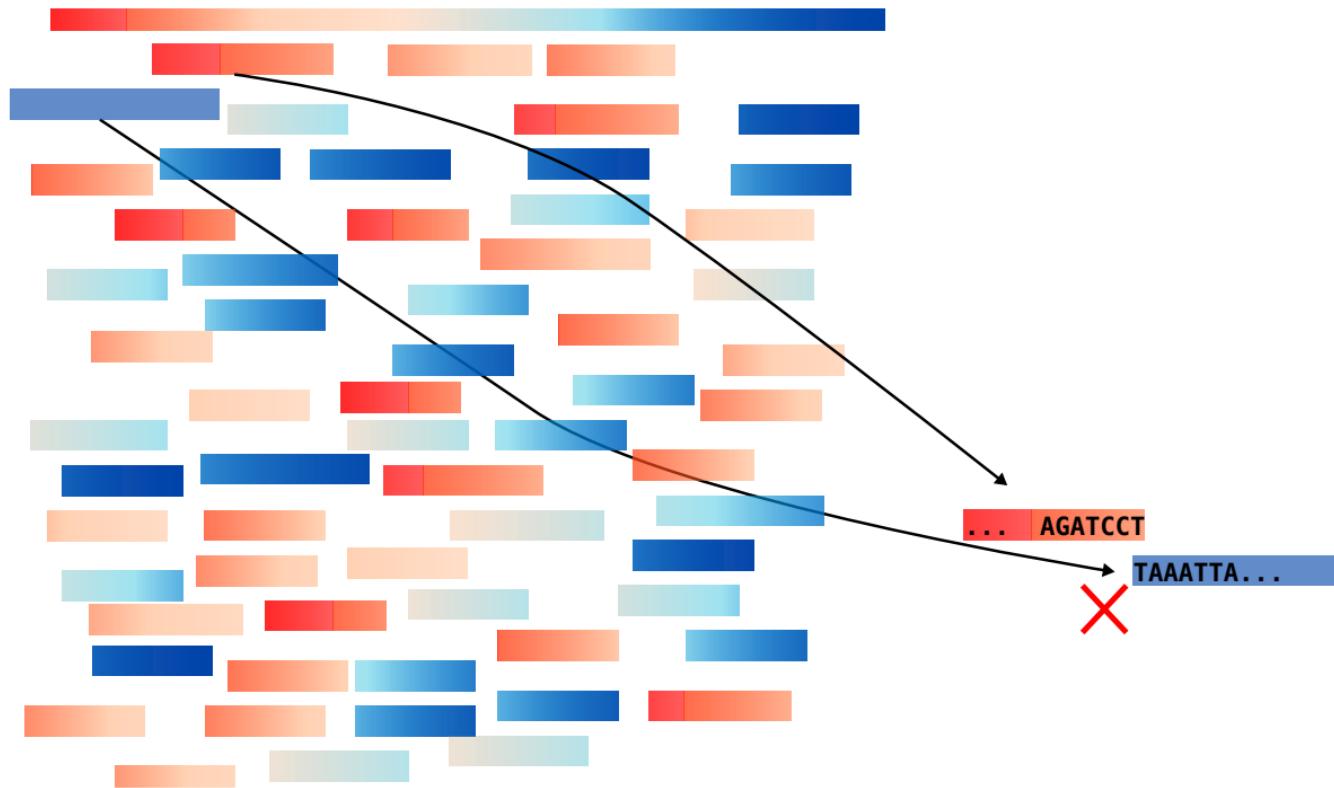
- Check all reads for overlaps (ii)



- **Most cases**

No overlap

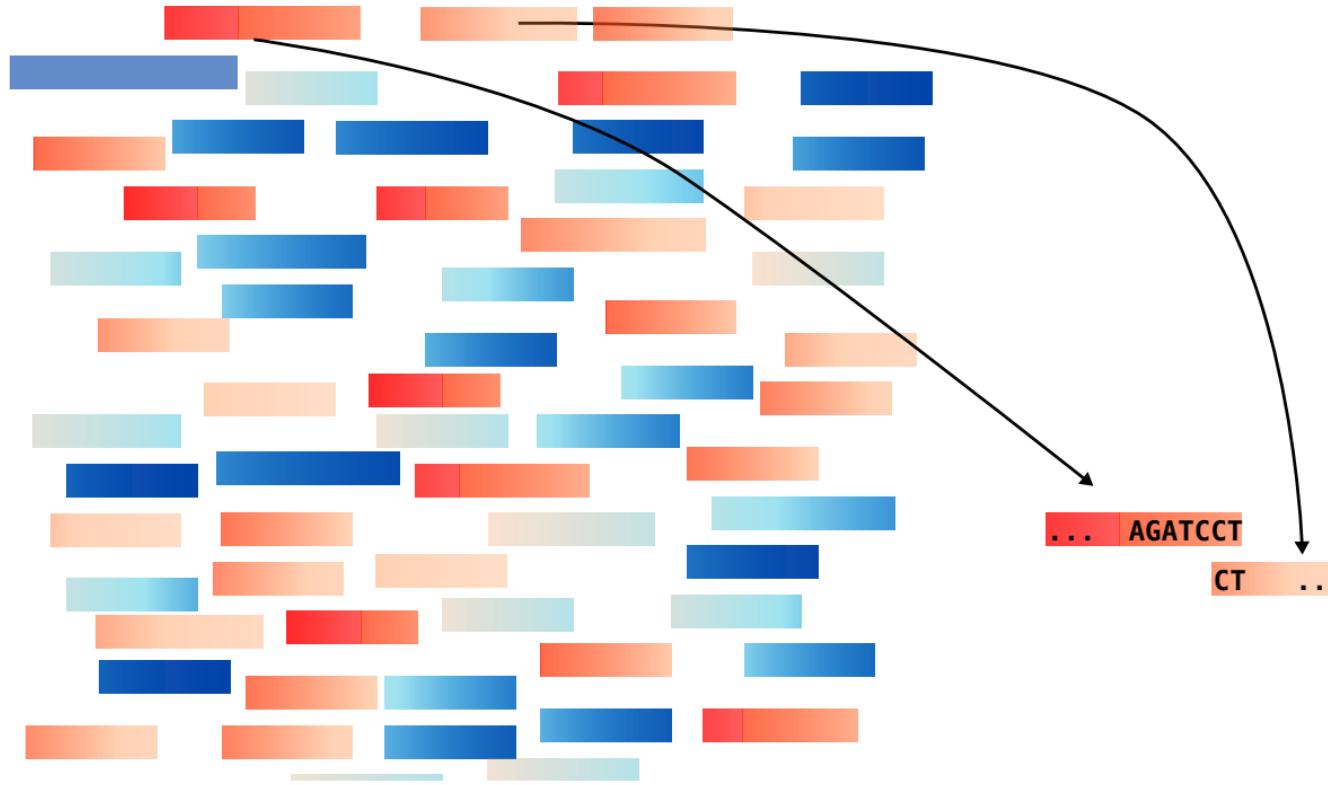
- Most cases (ii)



- **Small overlaps**

Can happen “by chance”

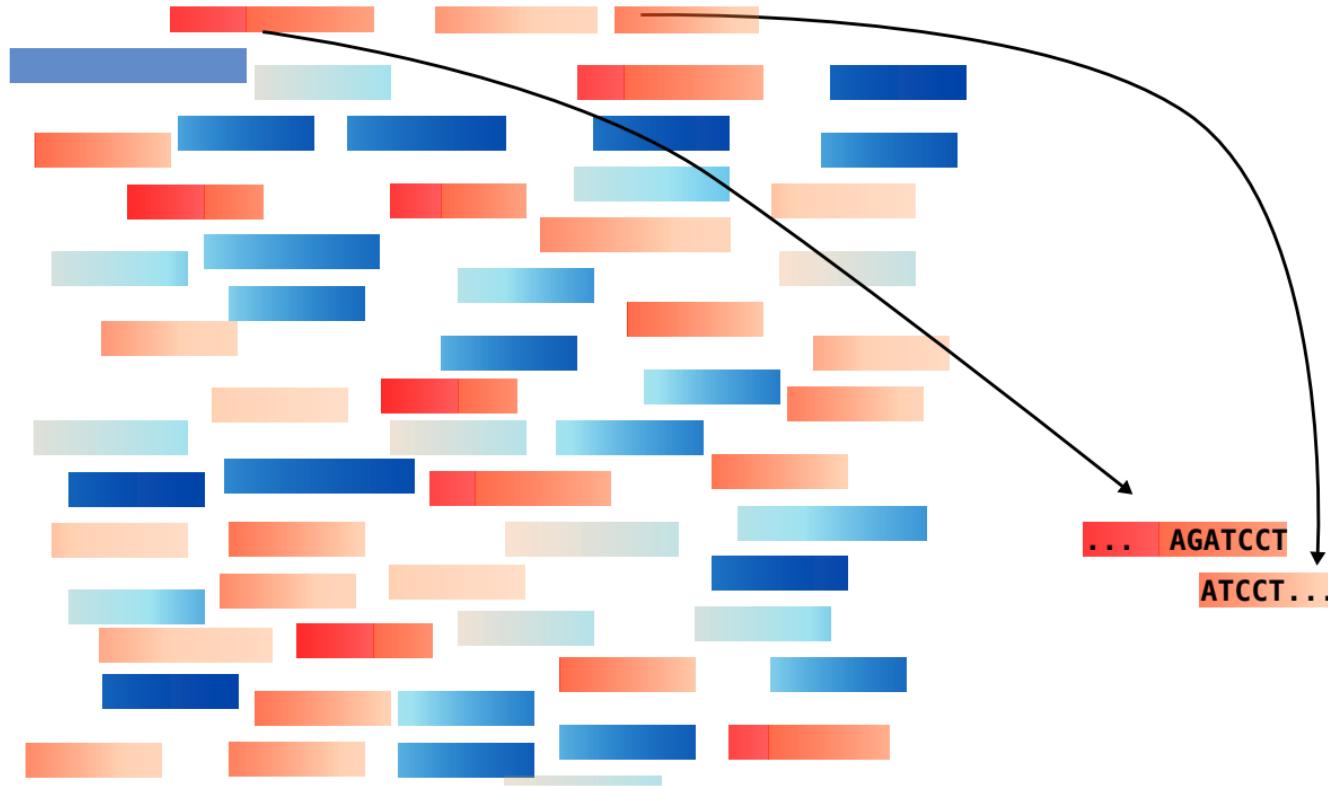
- Small overlaps (ii)



- **Longest overlaps**

We are more confident in longer overlaps

- Longest overlaps (ii)

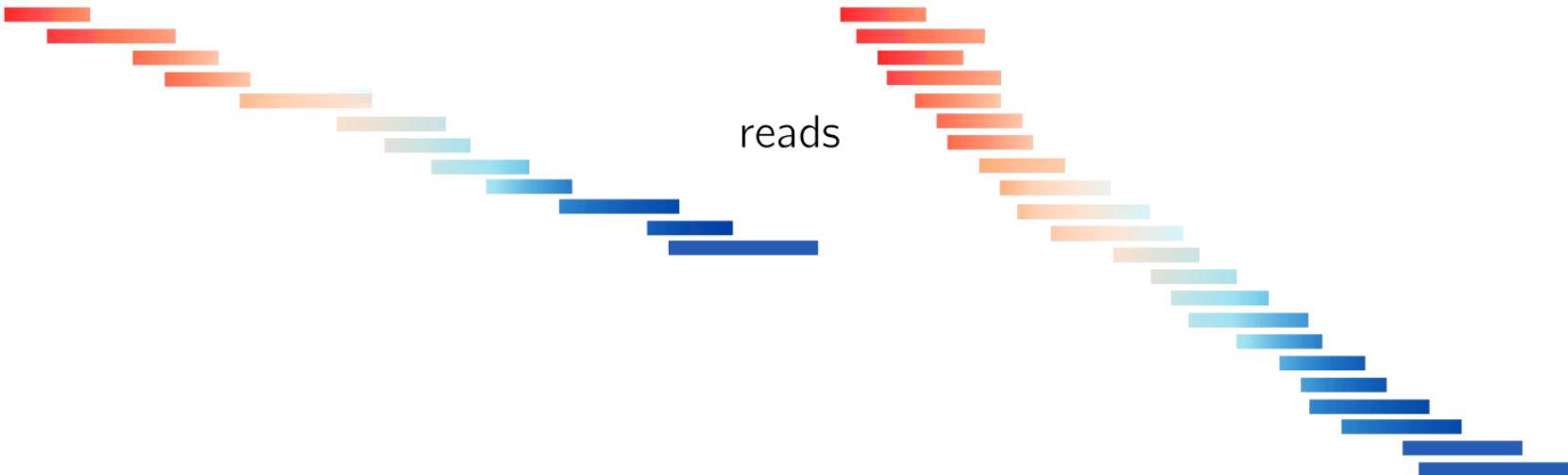


- Higher depth, longer overlaps

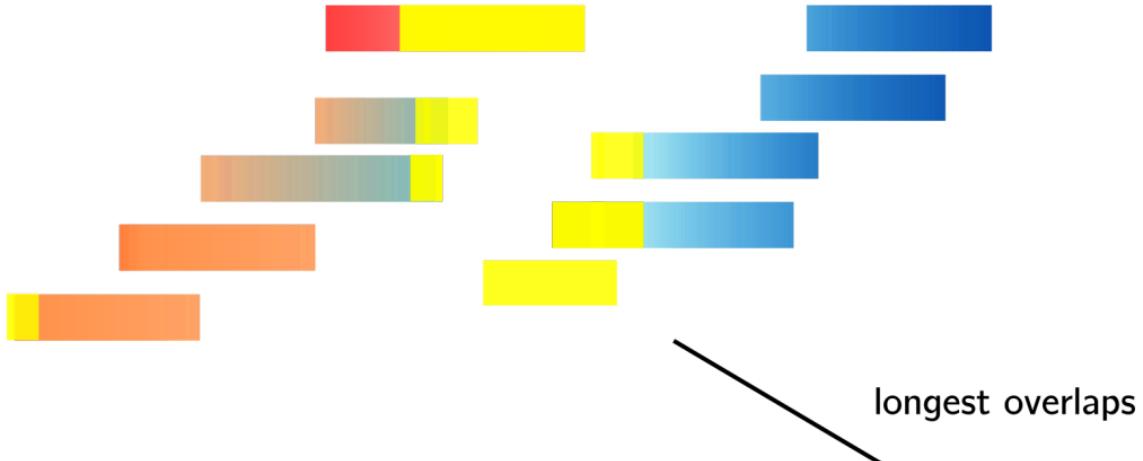
genome



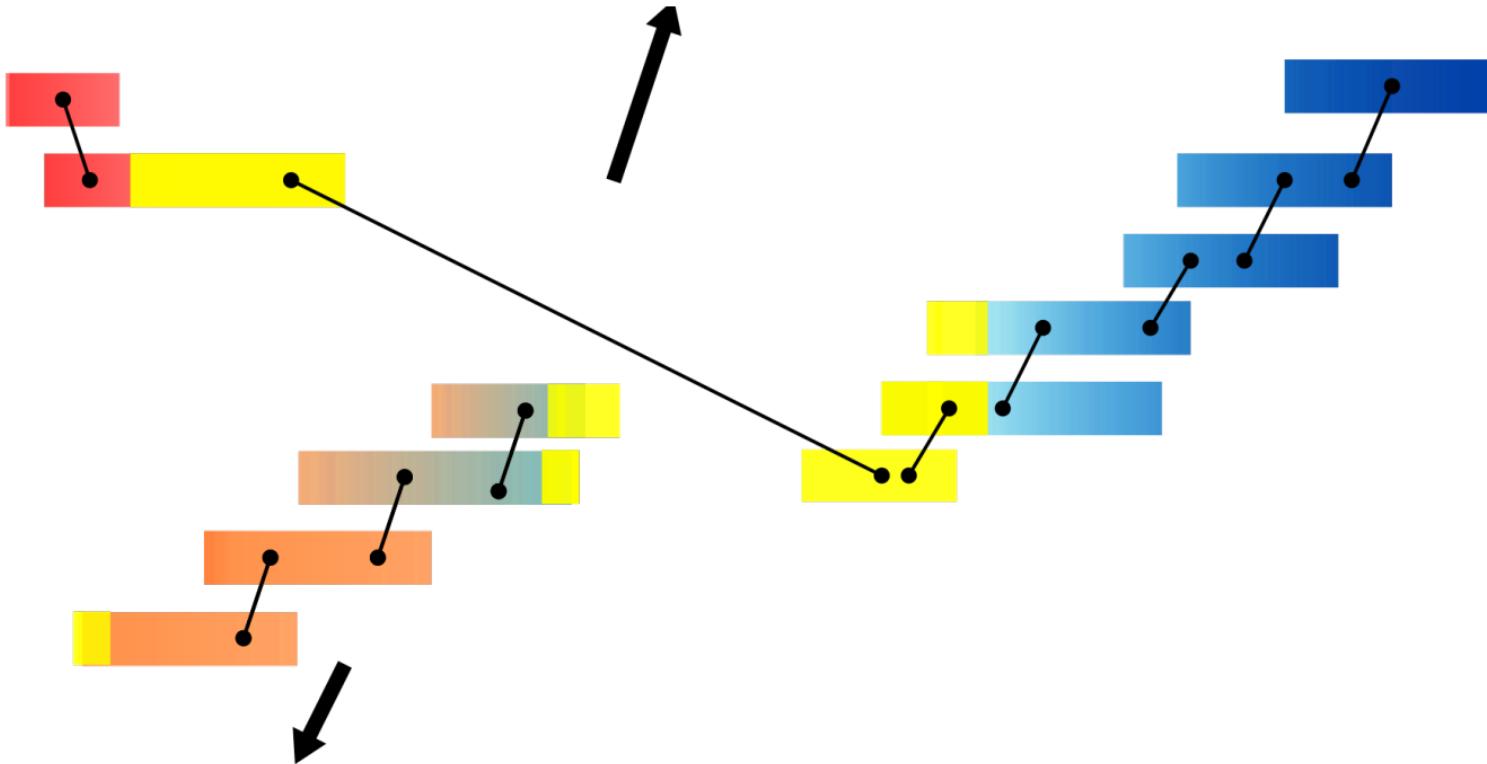
reads



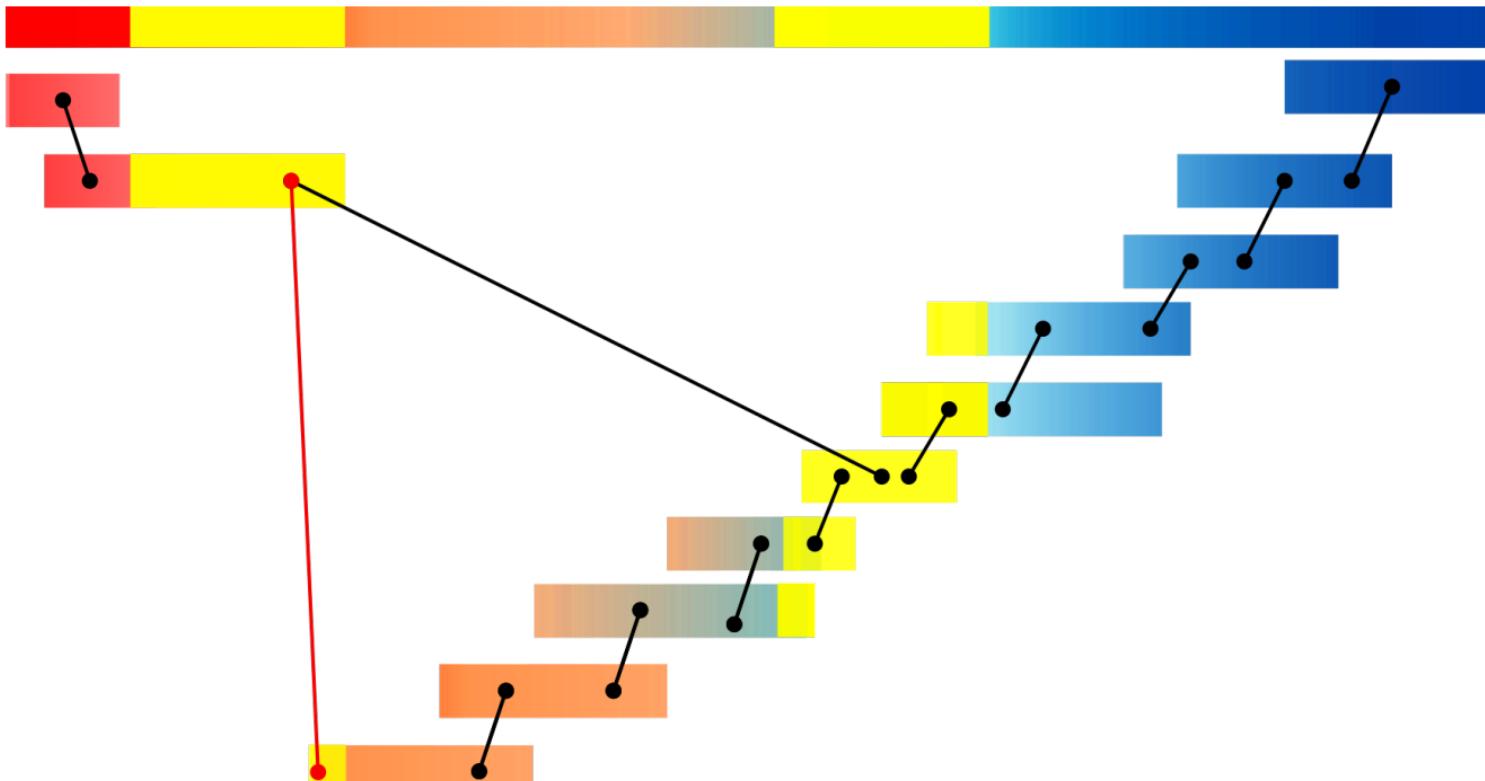
- Something weird happened



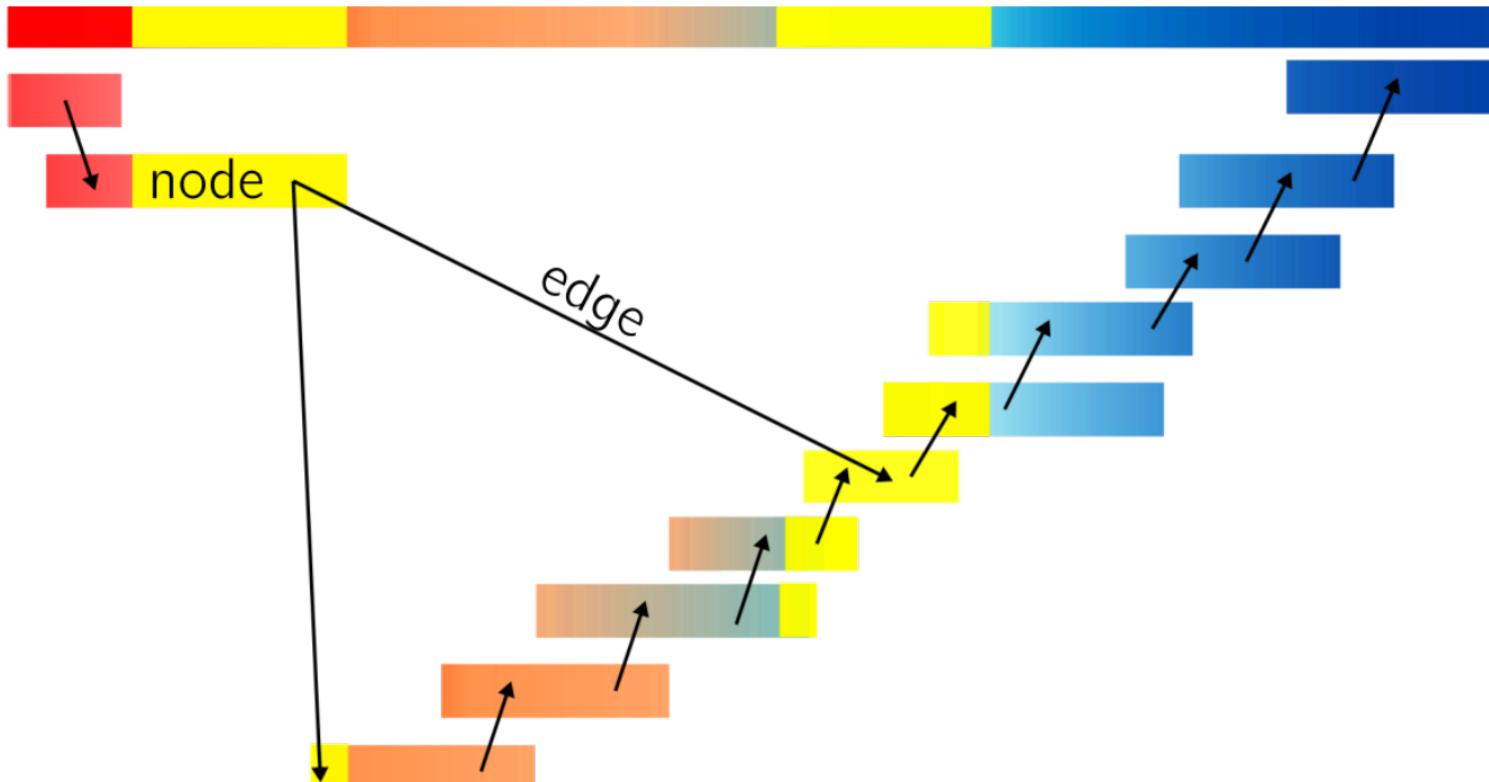
- All longest overlaps



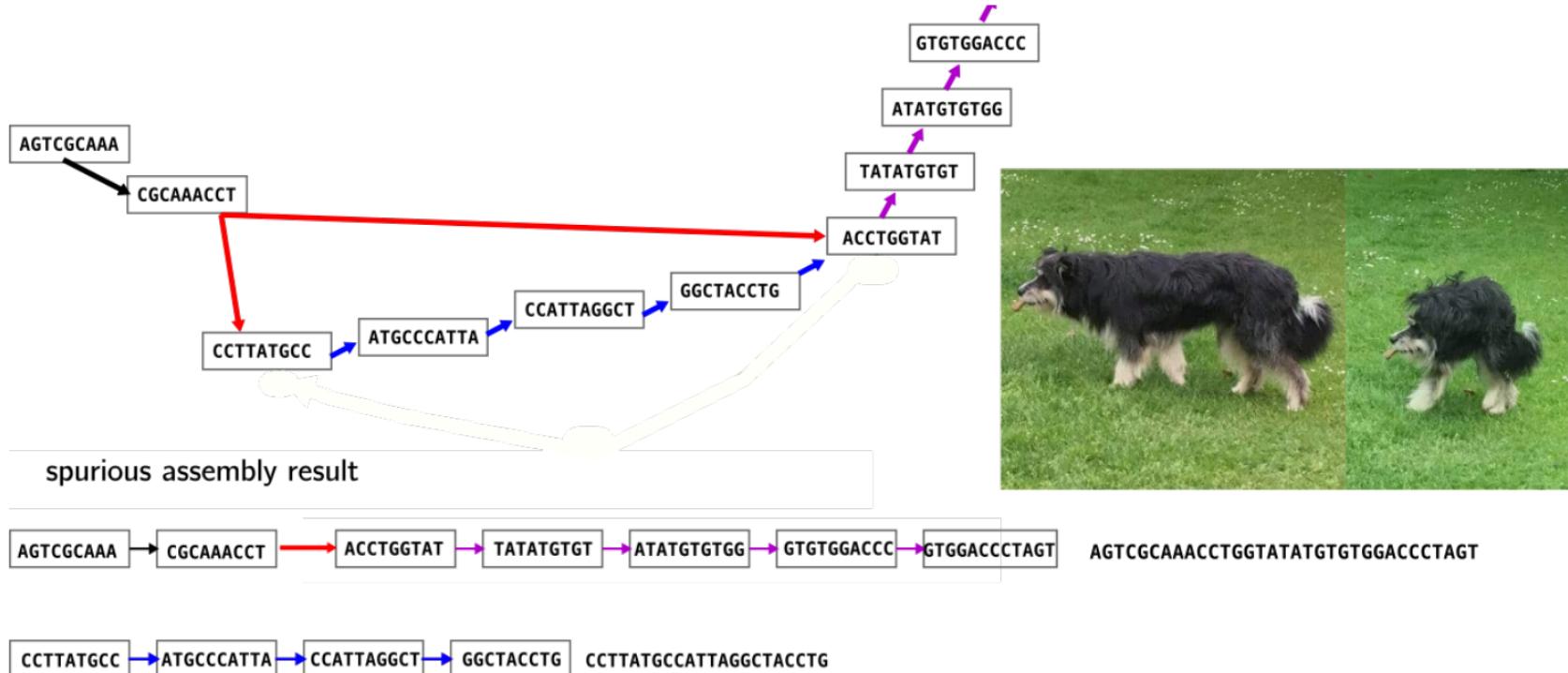
- Take into account other overlaps?



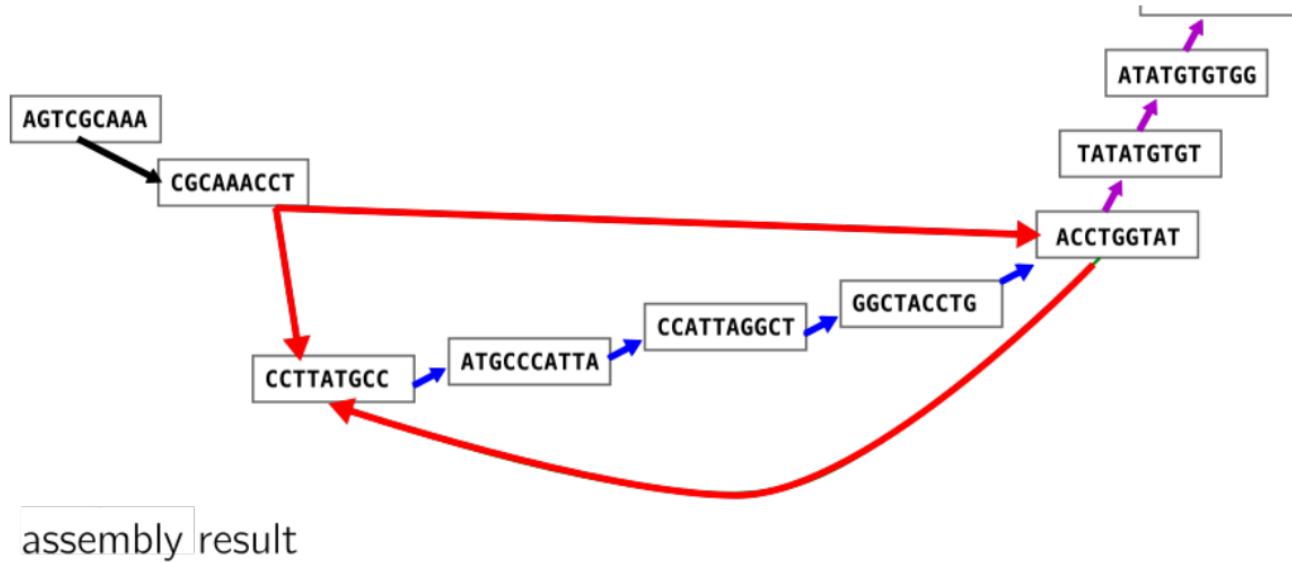
- Look, a graph!



- Unsafe paths in an overlap graph



- Safe paths in an overlap graph

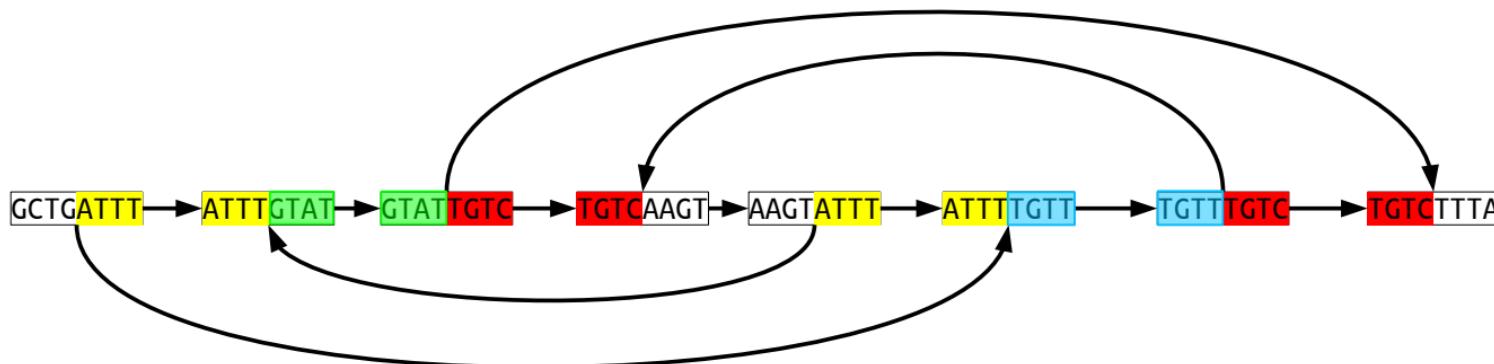


- Multiple repeats

Reads:

GCTGATT  
ATTTGTAT  
GTATTGTC  
TGTCAAGT  
AAGTATT  
ATTTTGTT  
TGTTTGTC  
TGTCTTTA

Overlap graph:



- First solution

Reads:

GCTGATT

ATTTGTAT

GTATTGTC

TGTCAAGT

AAGTATT

ATTTTGTT

TGTTTGTC

TGTCTTA

Overlap graph:



Possible assemblies:

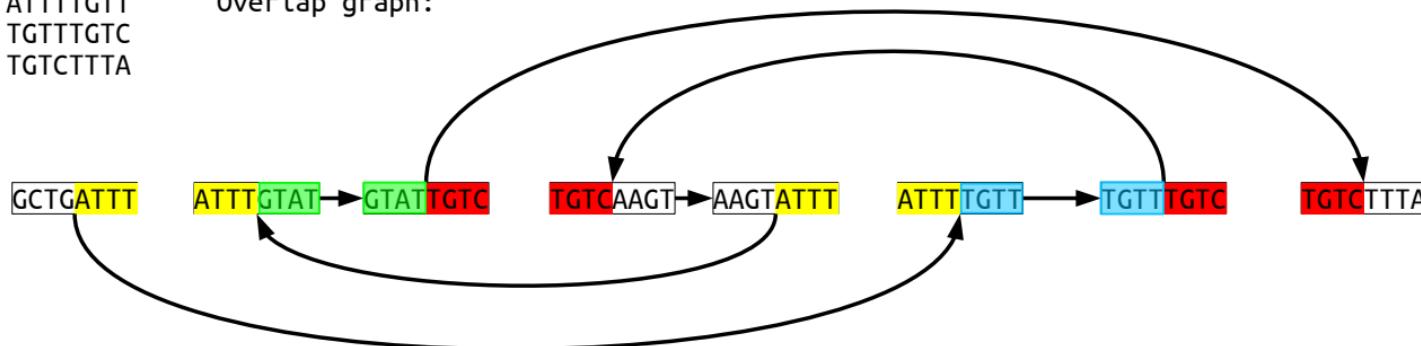
GCTGATTGTATTGTCAAGTATTTTGTTTGTCCTTA

## • Second solution

Reads:

GCTGATT  
ATTTGTAT  
GTATTGTC  
TGTCAAGT  
AAGTATT  
ATTTTGTT  
TGTTTGTC  
TGTCTTTA

Overlap graph:



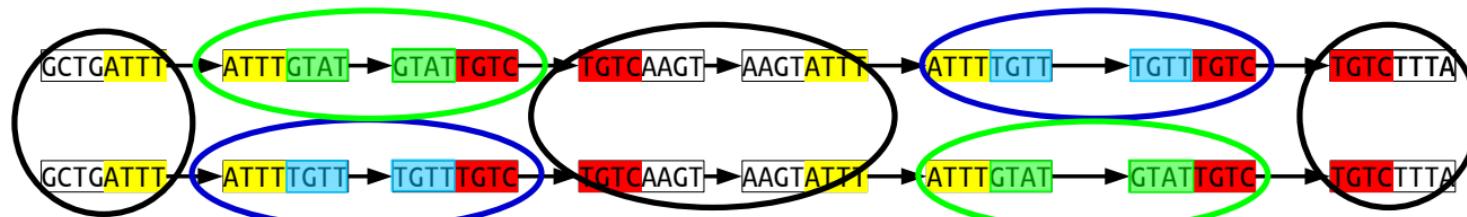
Possible assemblies:

GCTGATT<sub>1</sub> GTAT<sub>2</sub> TGTCAAGT<sub>3</sub> ATT<sub>4</sub> TGT<sub>5</sub> TGTCTTTA  
GCTGATT<sub>1</sub> TGT<sub>2</sub> TGTCAAGT<sub>3</sub> ATT<sub>4</sub> GTAT<sub>5</sub> TGTCTTTA

**Those two solutions are indistinguishable**

- Parsimonious solution: do not assemble

Possible assemblies:



Genome pieces:

GCTGATT | ATTT GTAT TGTC | TGTC AAGT ATT | ATTT TGTT TGTC |

Repeats lead to the fragmentation of the assembly

Genomes pieces that make **con<sup>\*</sup>sensus** across the different solutions are called Con<sup>\*</sup>tigs

- Do we expect many repeats?

Probability to have NO repeated word of size 31 in a 5 megabases genome

body

From [https://en.wikipedia.org/wiki/Birthday\\_problem](https://en.wikipedia.org/wiki/Birthday_problem)

- The burden of assembly: genomic repeats

Amount of repeats larger than a given size in *E. coli* genome

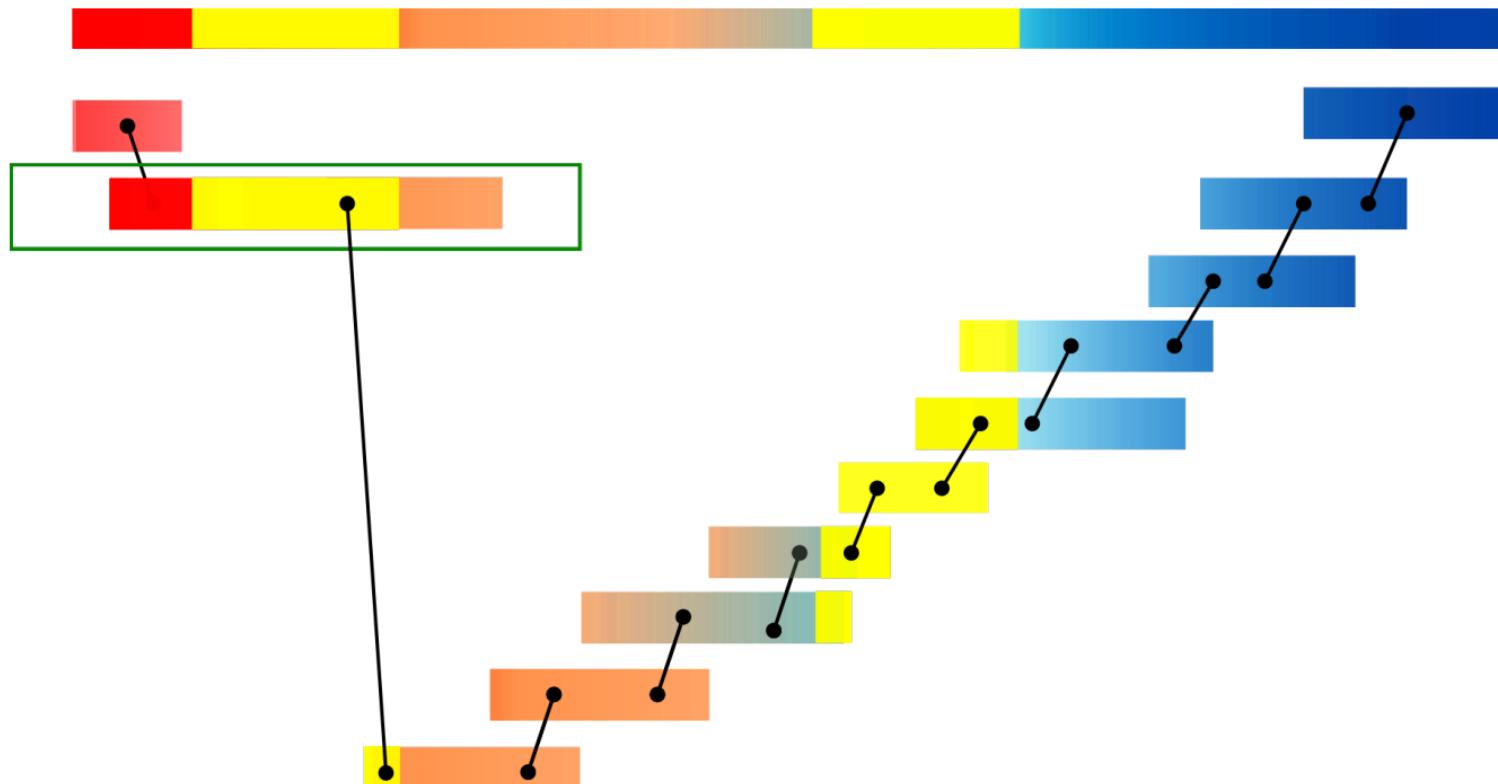
body

Genomic repeats are NOT random events

- With longer reads

Reads longer than the repeat “solve” it

- With longer reads (ii)



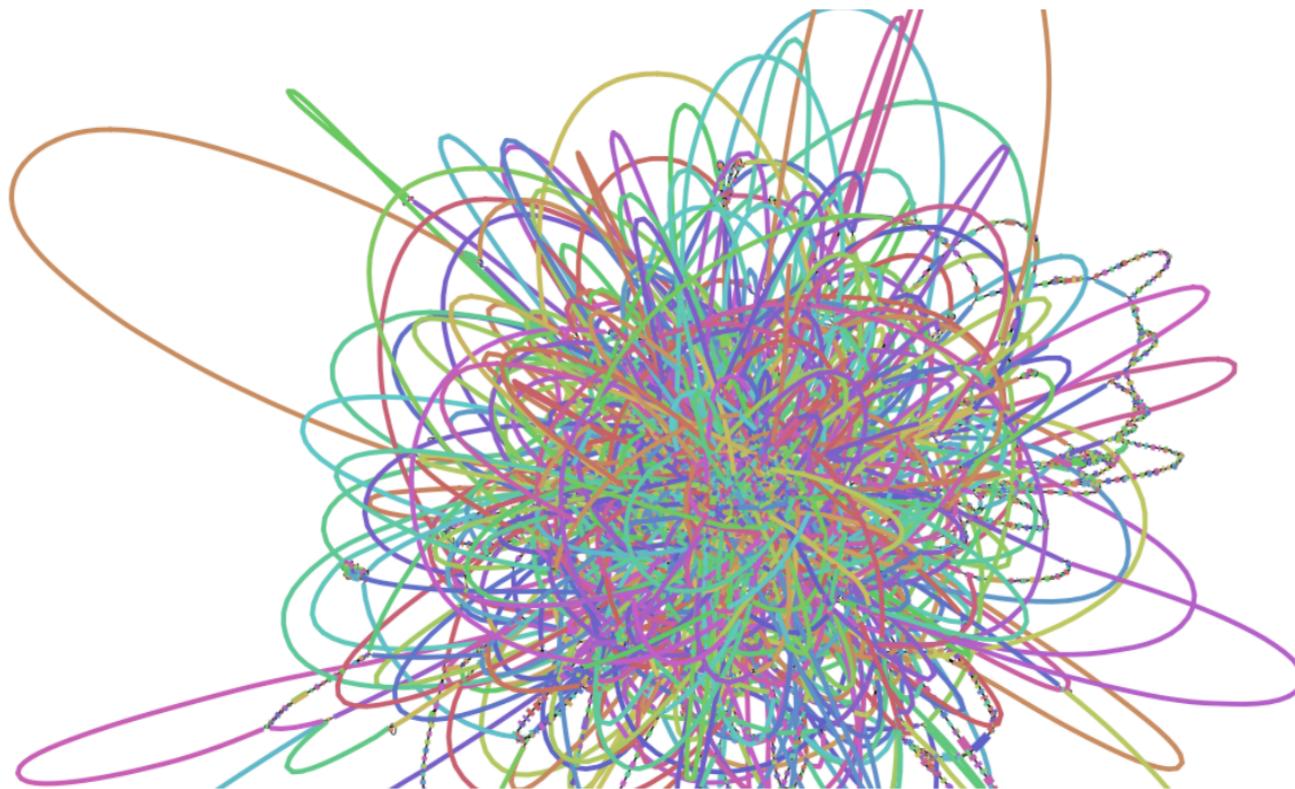
- With longer reads (iii)

The graph becomes trivial to traverse

- Read length matters

Read size=21

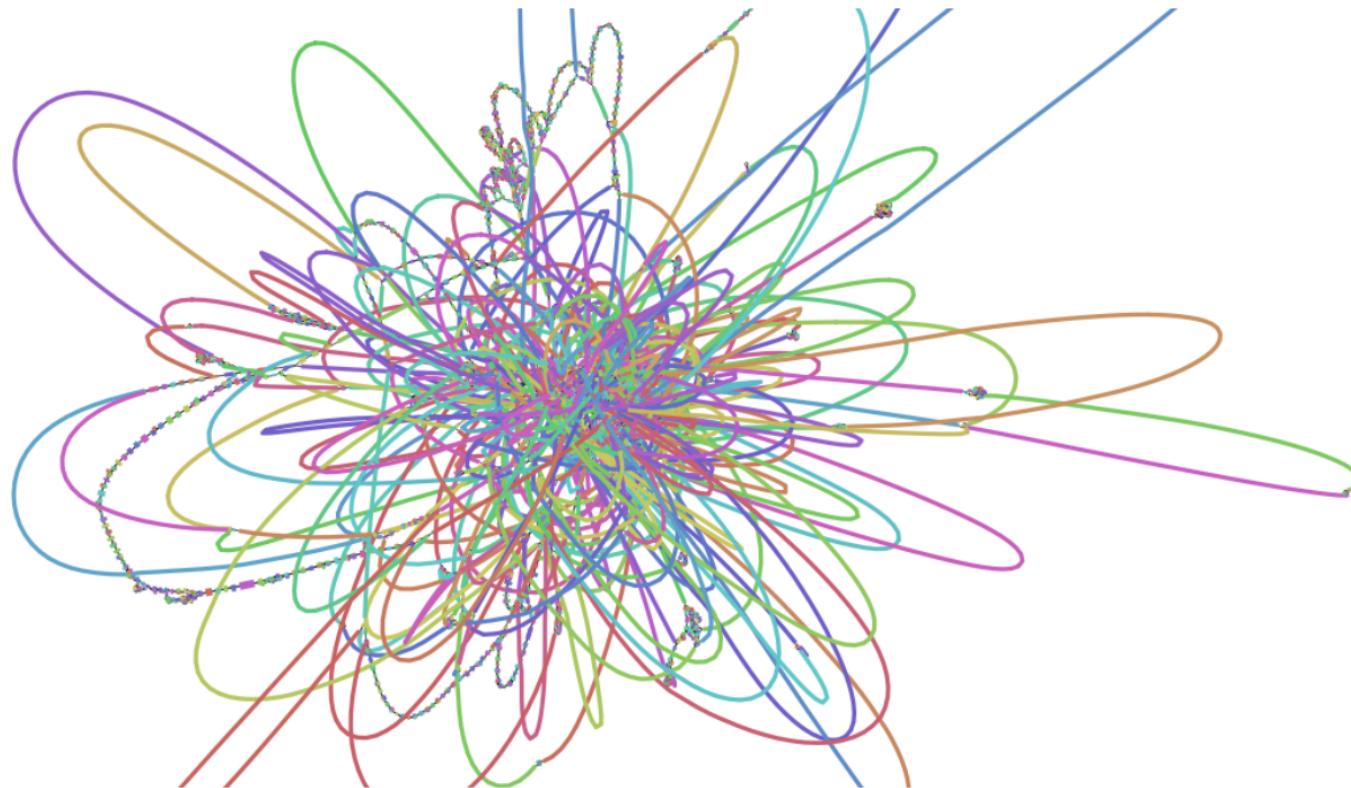
- Read length matters (ii)



- Read length matters

Read size=31

- Read length matters (ii)



- Read length matters

Read size=63

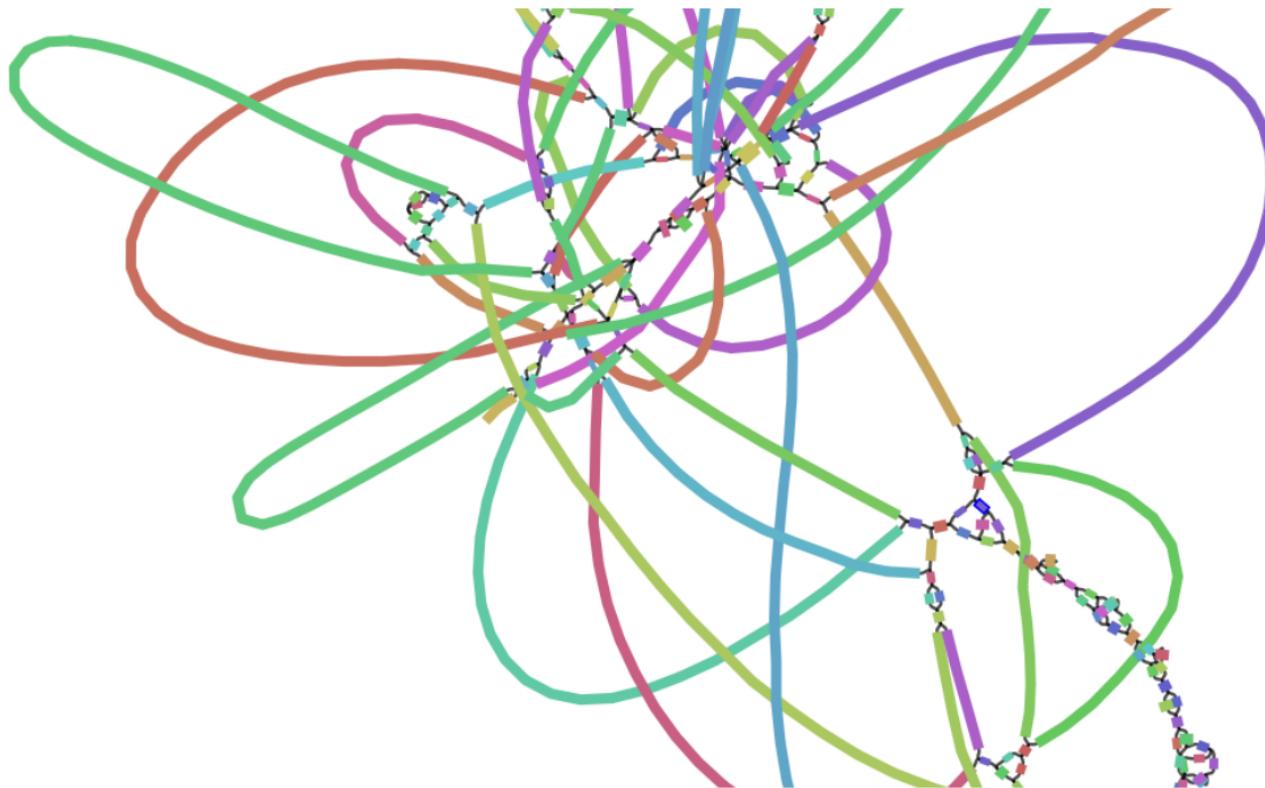
- Read length matters (ii)



- Read length matters

Read size=255

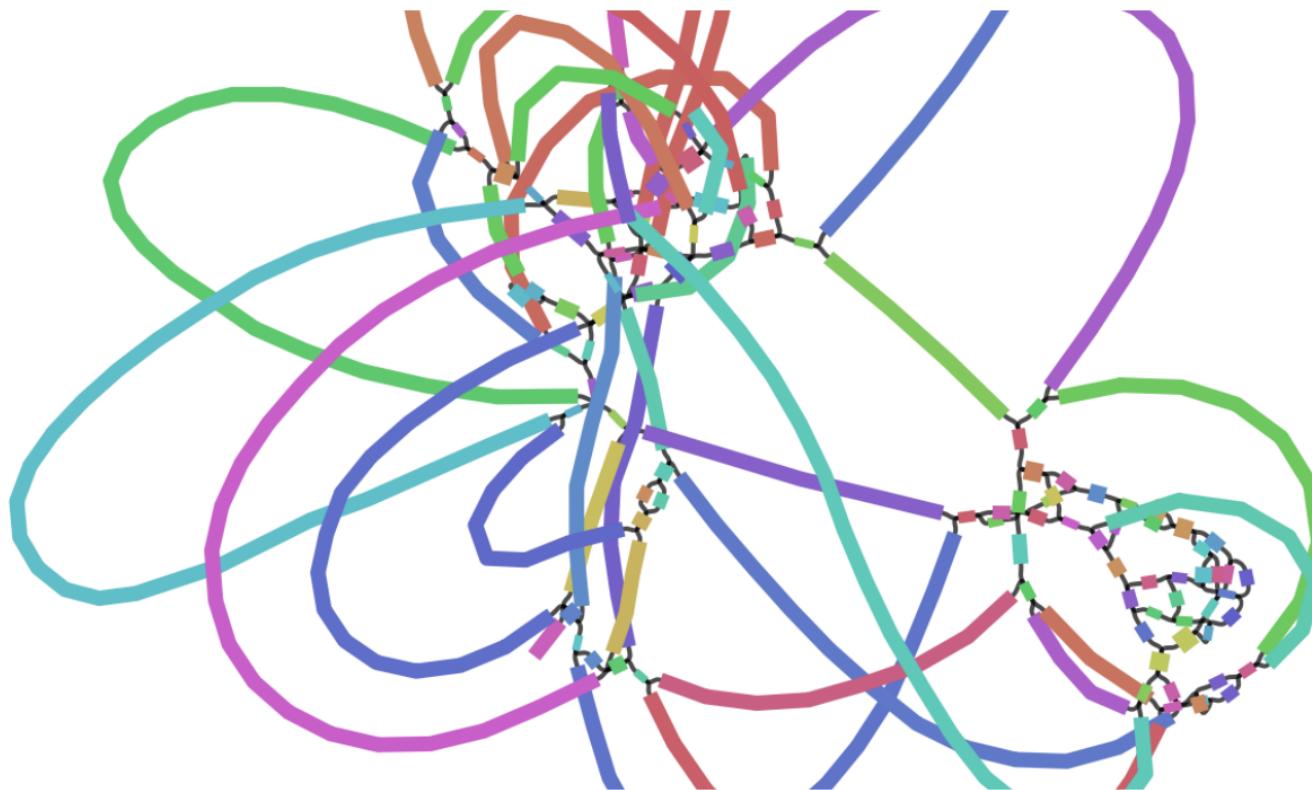
- Read length matters (ii)



- **Read length matters**

Read size=500

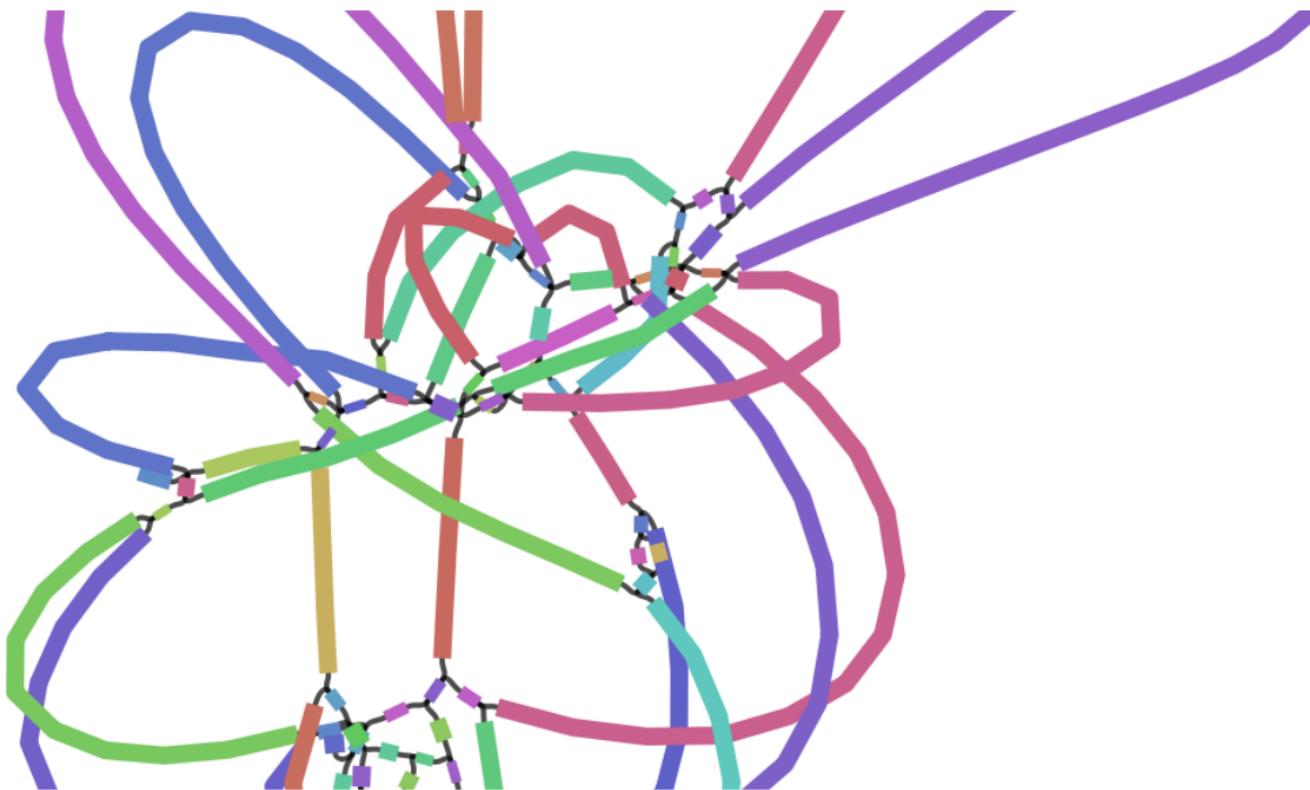
- Read length matters (ii)



- **Read length matters**

Read size=1000

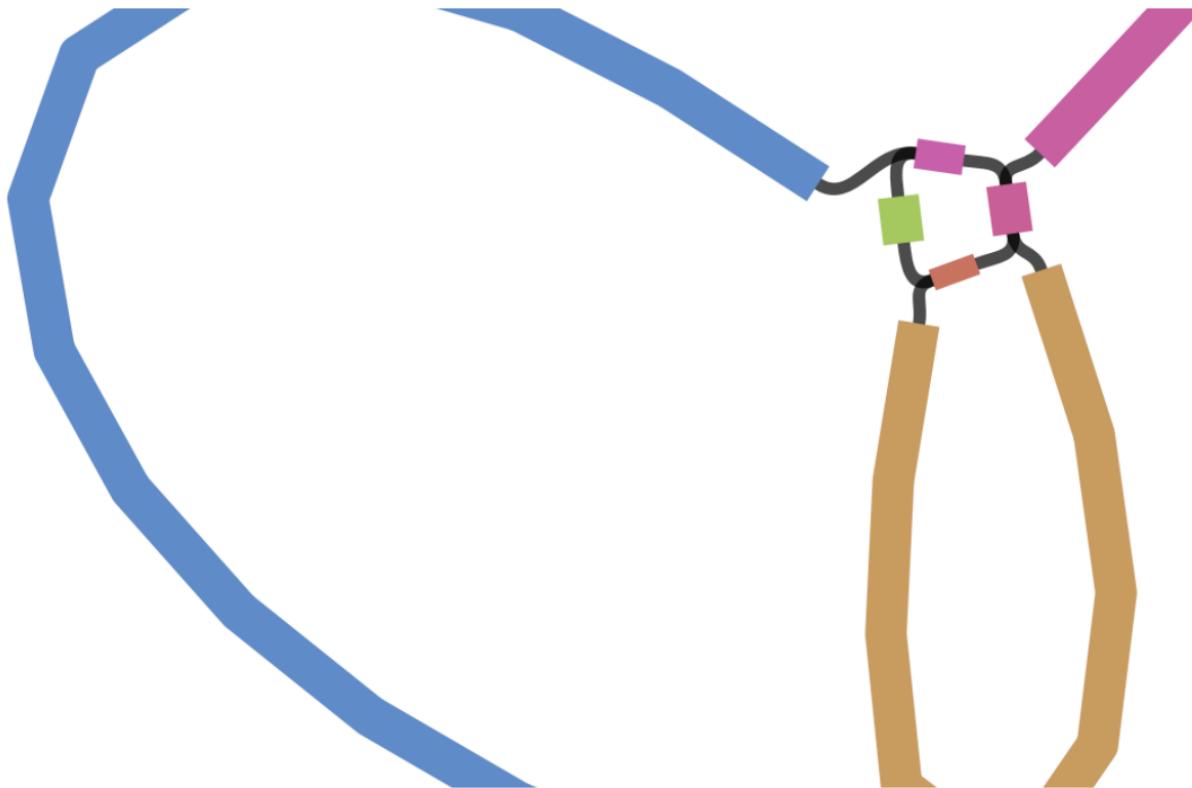
- Read length matters (ii)



- **Read length matters**

Read size=2000

- Read length matters (ii)



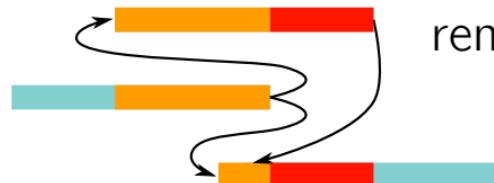
- Overlap graph simplifications



remove small overlaps



remove node inclusions



remove dominated  
overlaps



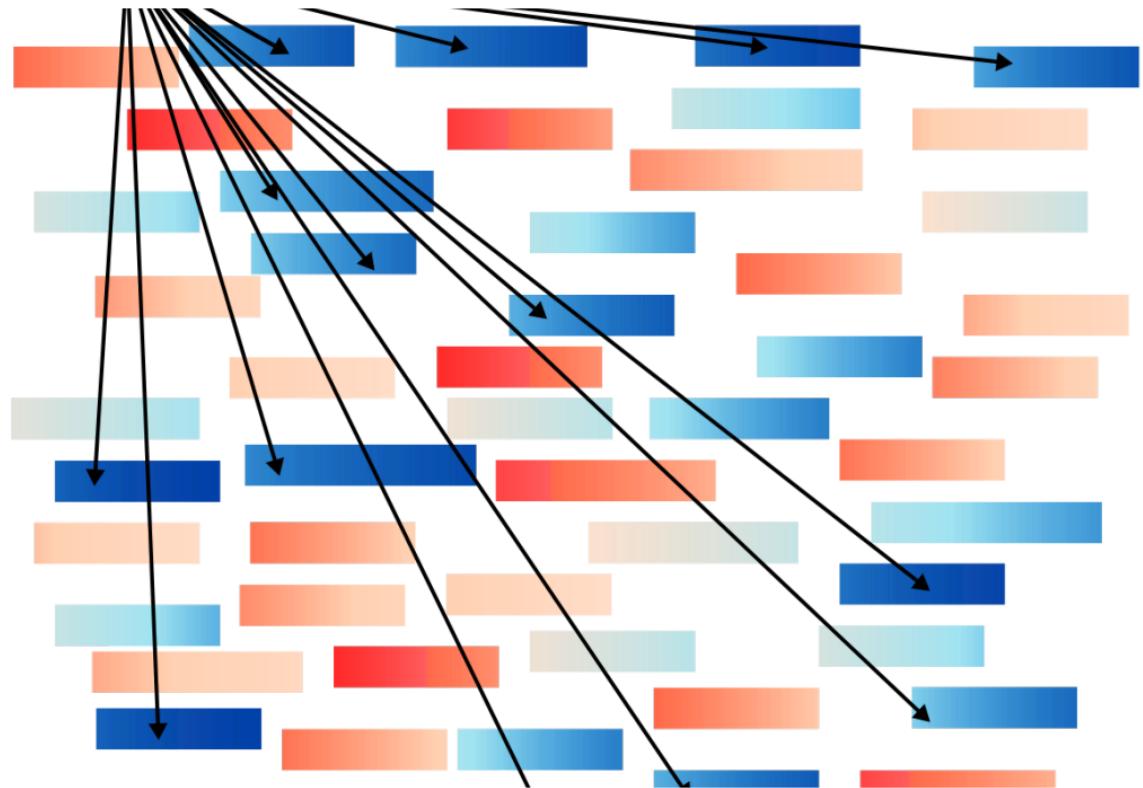
- **First (and most important) checkpoint**

- Assembly orders reads using overlaps; longer overlaps are **generally** better.
- Multiple possible overlaps necessitate graphs for structuring information.
- Repeats longer than reads result in fragmented assembly (contigs).

- Compute overlaps

Detecting overlaps means a lot of comparisons

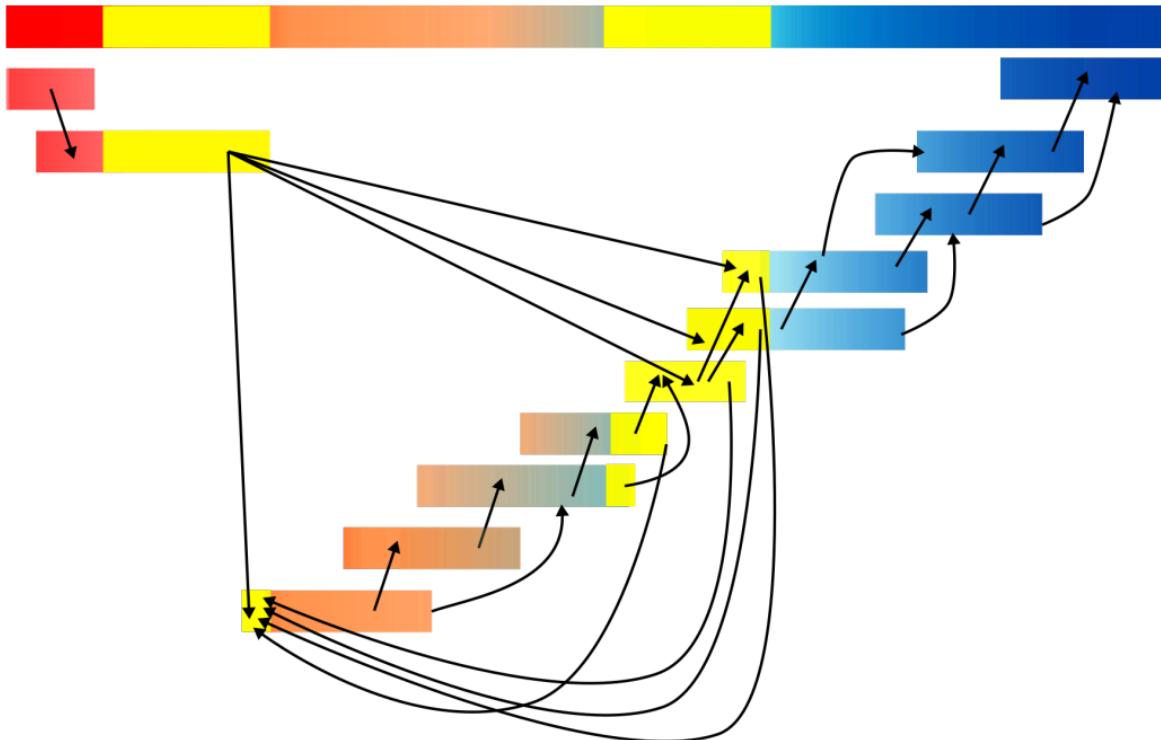
- Compute overlaps (ii)



- Compute overlaps

Even considering only the long overlaps means a lot of comparisons

- Compute overlaps (ii)



- Overlap graph burden: number of reads

$\frac{n(n-1)}{2} = O(n^2)$  possible overlaps for  $n$  reads

# Reads	# Overlaps
1000	499,500
10,000	50 million
100,000	5 billion
1 million	500 billion
10 million	50 trillion...

We have to be efficient and focus on “relevant” overlaps

- Overlap graph burden: number of overlaps

For each base of the genome:

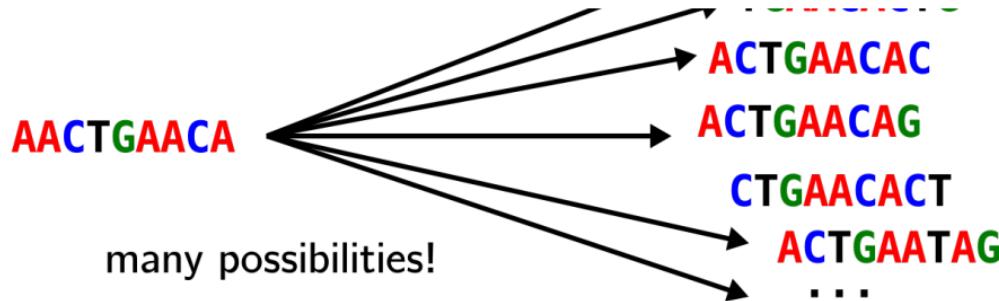
Read depth	Overlaps depth
10	100
20	400
50	2,500
100	10,000

The amount of overlaps is not linear

Linear: 2X data 2X time

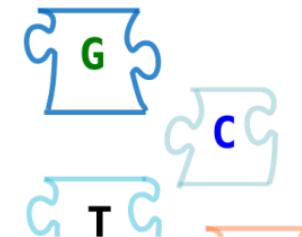
Quadratic: 2X data 4X time

- Another idea



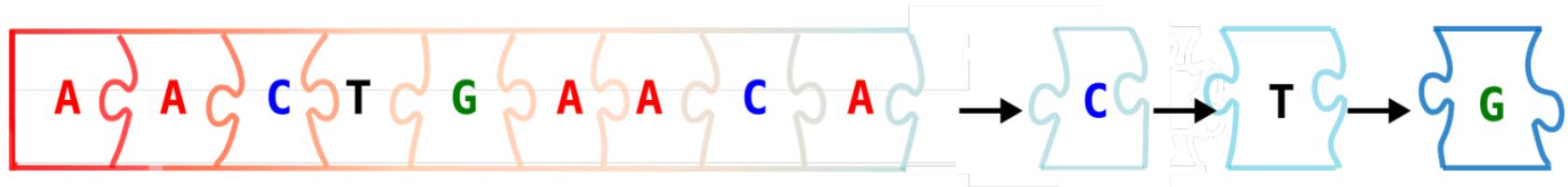
## NEW SOLUTION !

instead, from the context, find out the next nucleotide

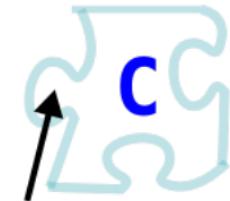


- Another idea

add the nucleotides one by one

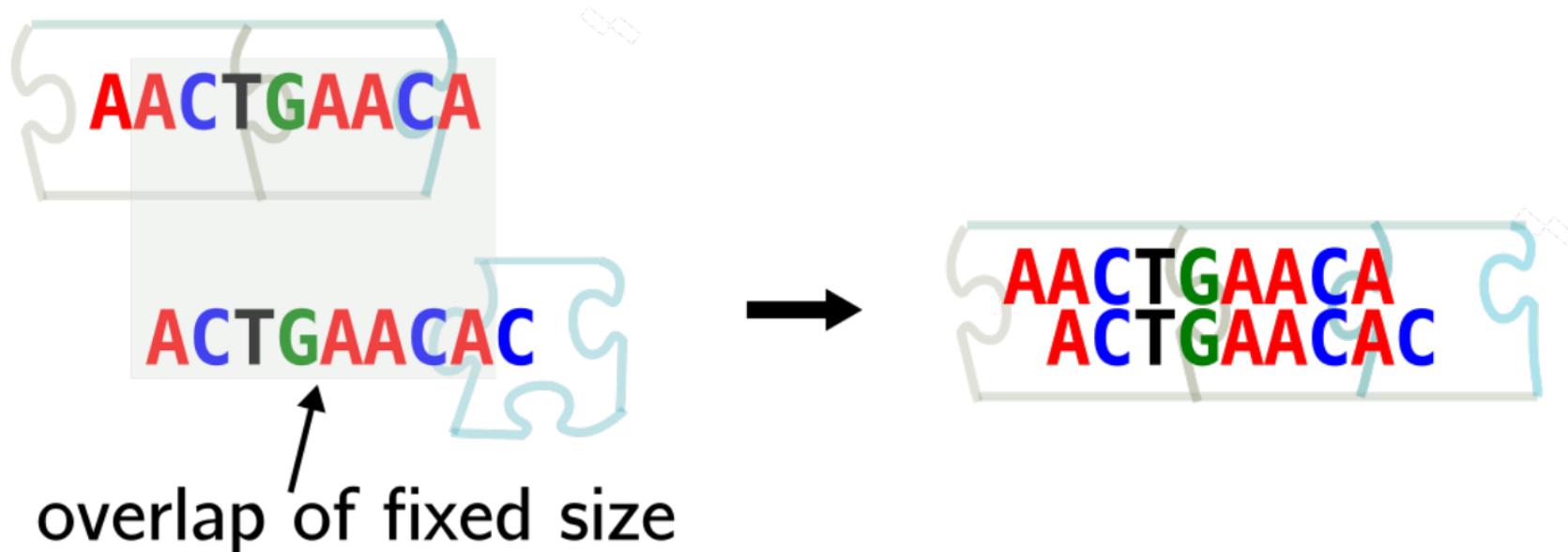


- Context



context to join the next base?

- Context



- Assembly

The diagram shows four DNA sequence fragments arranged vertically:

- Fragment 1: AACTG (red)
- Fragment 2: ACTG (black)
- Fragment 3: CTGAACACT (blue)
- Fragment 4: TGAACACTG (green)

Arrows point from the overlapping regions between adjacent fragments to the right, indicating the assembly process where overlapping reads are joined together.

- The de Bruijn graph



## De Bruijn graph

Kmer=node



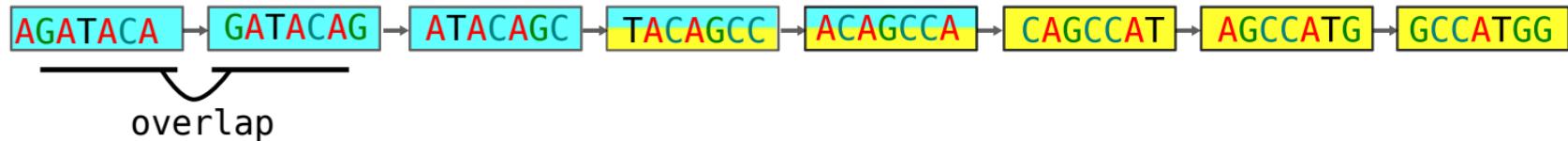
AGATA + G + C + C + A

- de Bruijn graph assembly

Overlapping reads

AGATA**CAGCCA**  
**TACAGCCATGG**

De Bruijn graph



Resulting sequence

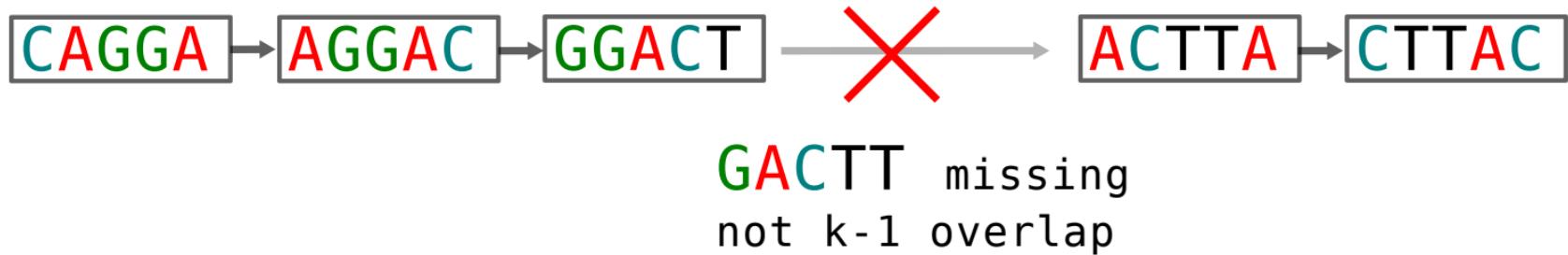
AGATA**CAGGCCATGG**

## • Why bother with k-mers?

in my graph,  $k$ -mer size = read size



- de Bruijn graphs limitation 1: Fixed overlaps



GGACT and ACTTA overlap is only of size 3 !

- de Bruijn graphs limitation 1: Fixed overlaps (ii)

- Exercise 1: de Bruijn graph time!

Reads

GCCATGGGTTT

TACAGGCCATGG

AGCCATGGGTT

GCCATGGGTTT

AGCCATGGGTT

ACAGGCCATGGG

GATACAGGCCAT

ATACAGGCCATG

CATGGGTTTAA

CAGCCATGGGT

GATACAGGCCAT

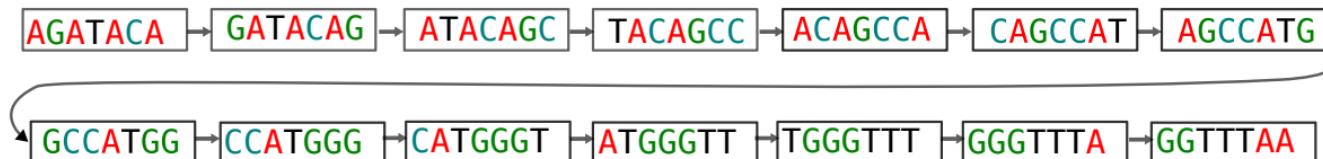


Hint: Use 7-mers

- Exercise 1: Solution

GATACAGCCAT  
ATACAGCCATG  
TACAGCCATGG  
ACAGCCATGGG  
ACAGCCATGGG  
CAGCCATGGGT  
AGCCATGGGTT  
GCCATGGGTTT  
GCCATGGGTTT  
CCATGGGTTTA  
CATGGGTTTAA

de Bruijn graph



- de Bruijn graphs abstract redundancy

GATAACAGCCAT

ATACAGGCCATG

TACAGGCCATGG

ACAGGCCATGGG

ACAGGCCATGGG

CAGGCCATGGGT

AGCCATGGGTT

GCCATGGGTTT

GCCATGGGTTT

CCATGGGTTTA

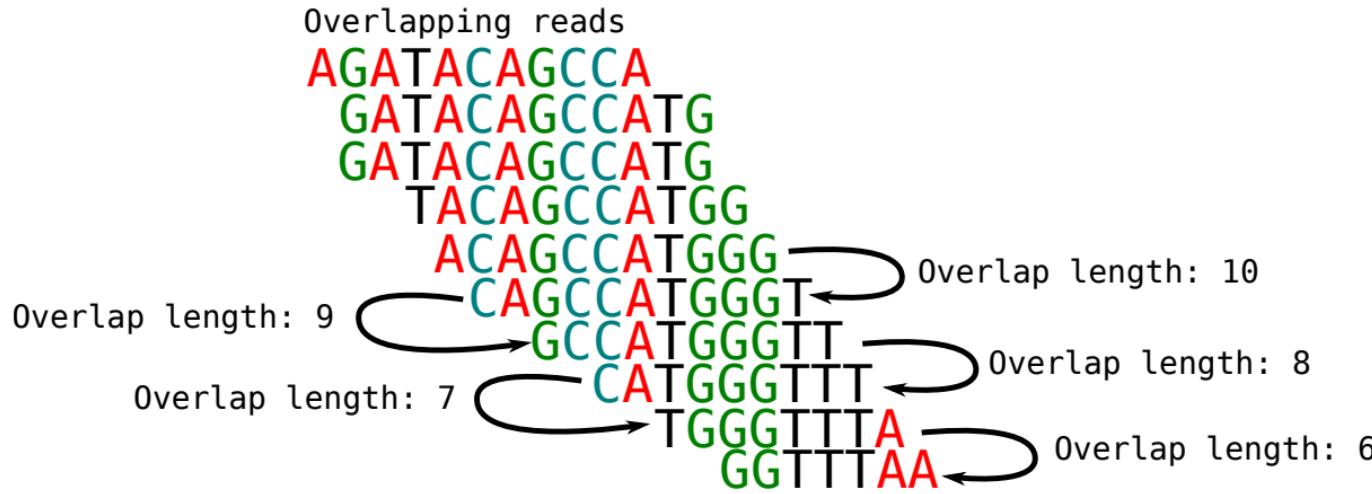
CATGGGTTAA

65 non distinct 7-mers in reads

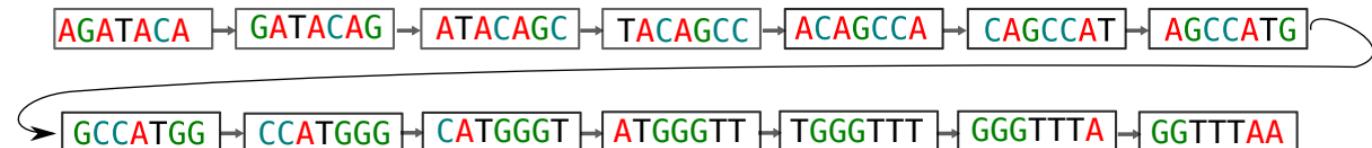
14 distinct 7-mers in the de Bruijn graph



- de Bruijn graphs only rely on  $k - 1$  overlaps



De Bruijn graph overlap length: 6



- Repeats in a de Bruijn graphs

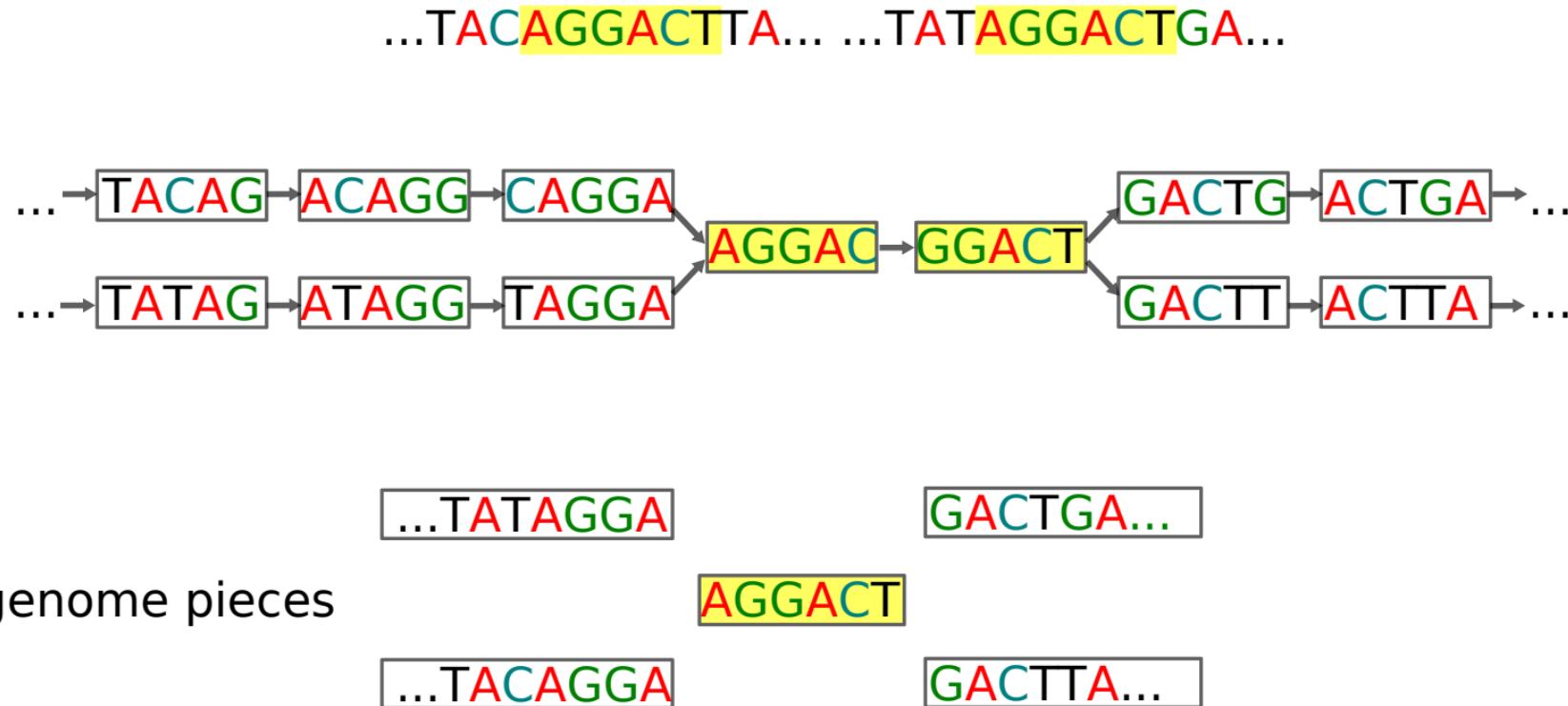
...TACAGGACTTA... ...TATAGGACTGA...



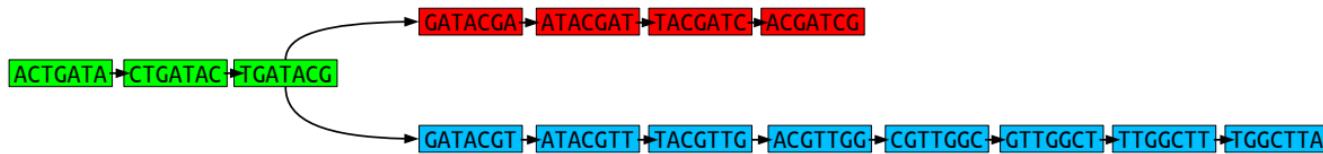
#v(0.4cm)

each  $k$ -mer appears only once in a de Bruijn graph

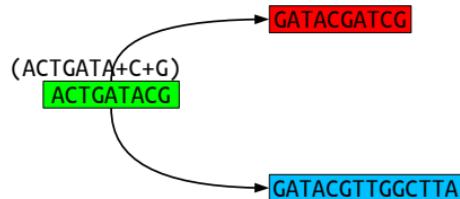
- de Bruijn graphs limitation 2: Repeats



- On the representation of de Bruijn graphs



Compacted De Bruijn graph:



Graphical representation (.gfa plot using Bandage):



- The boy is diploid!

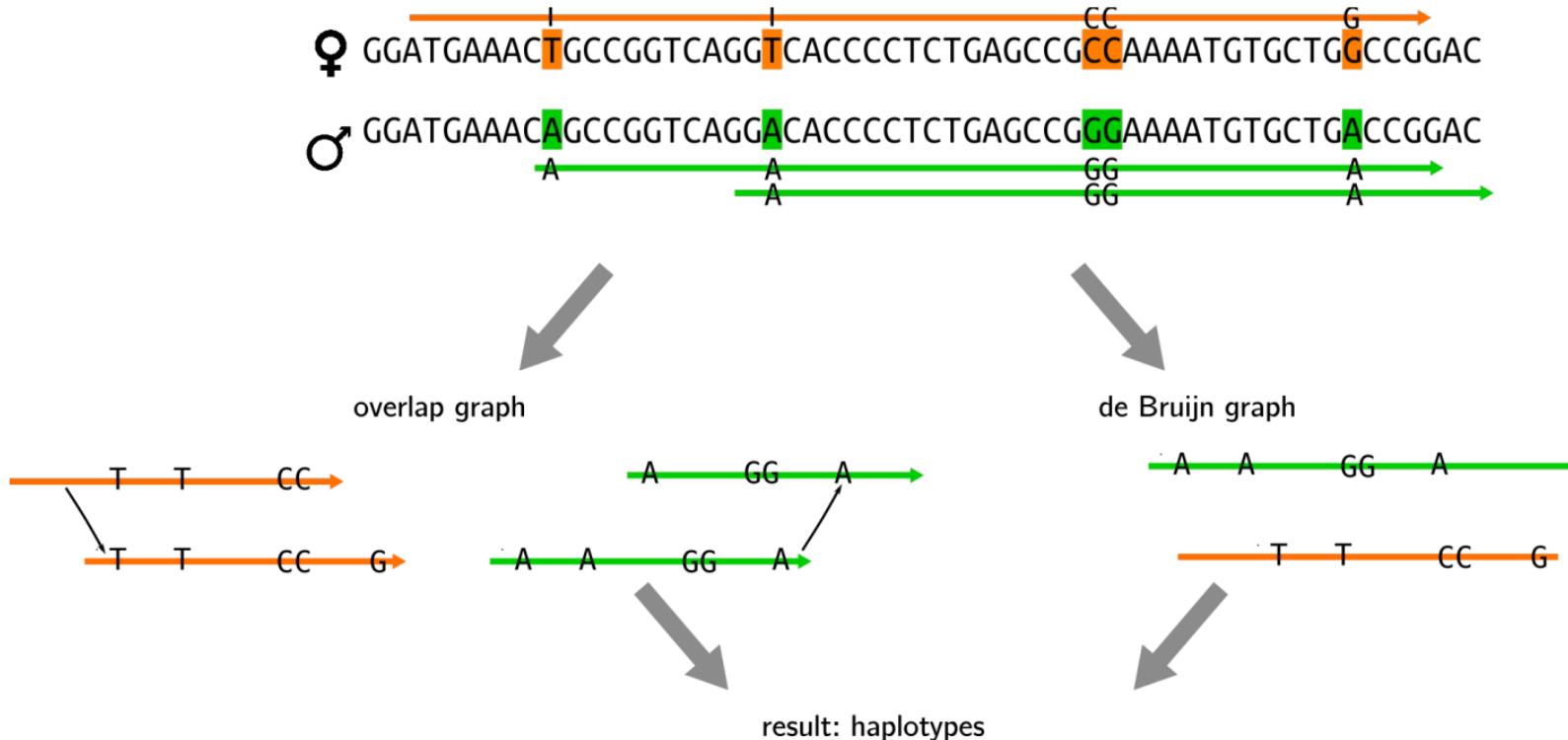
♀ GGATGAAACTGCCGGTCAGGTACCCCTCTGAGCCGCCAAAATGTGCTGGCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGCCAAAATGTGCTGACCGGAC



- The boy is diploid! (ii)

- Ploidy and very long reads



- Ploidy and very long reads (ii)

Haplotypes can be “phased”

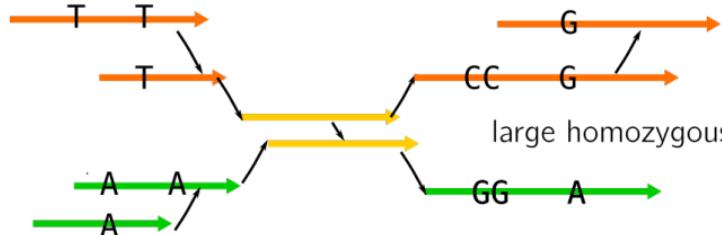
- Ploidy and very long reads (iii)

- Homozygous vs heterozygous regions

♀ GGATGAAACTGCCGGTCAGGTACACCCCTCTGAGCCGCCAAAATGTGCTGCCGGAC

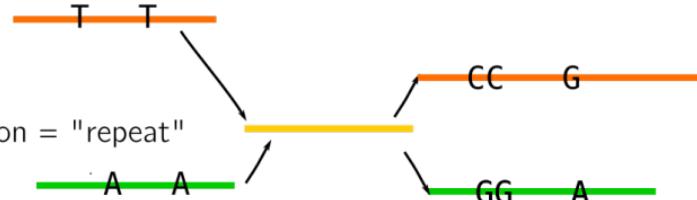
♂ GGATGAAACAGCCGGTCAGGAACACCCCTCTGAGCCGCCAAAATGTGCTGACCCGAC  
 ↓  
 A A A A GG GG A A

overlap graph



large homozygous region = "repeat"

de Bruijn graph

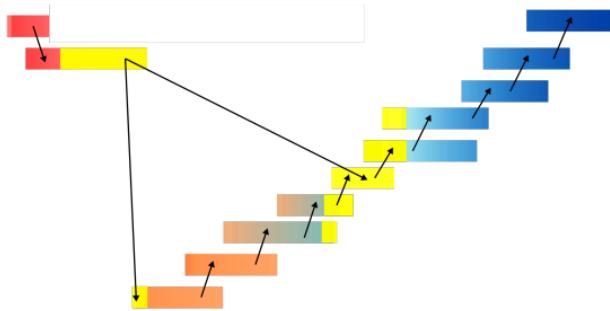


- **Homozygous vs heterozygous regions (ii)**

Assembly concession: assembly can be fragmented due to ploidy

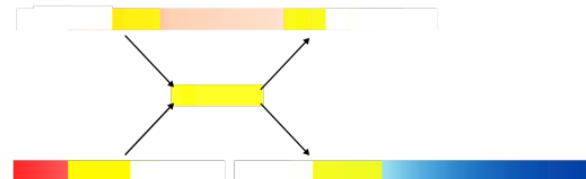
- Homozygous vs heterozygous regions (iii)

- Method checkpoint: de Bruijn graph versus overlap graph



quadratic growth with coverage

issue with repeats larger than reads



abstracts coverage

issue with repeats larger than  $k$

- Data checkpoint: results for the *long, perfect boy*



100kb region from the genome



haplotype 1  
haplotype 2

10 million reads → 1000 contigs



- Contigs can reach the chromosome's order of magnitude in length (megabases)
- Breaks due to large repeats
- Haplotypes can be partially reconstructed

- Second experiment: *noisy, super long boy's genome*

(only for the record, we actually don't have it)



Genome size  
1 billion bases



Reads  
1 million  
mean size 100kb

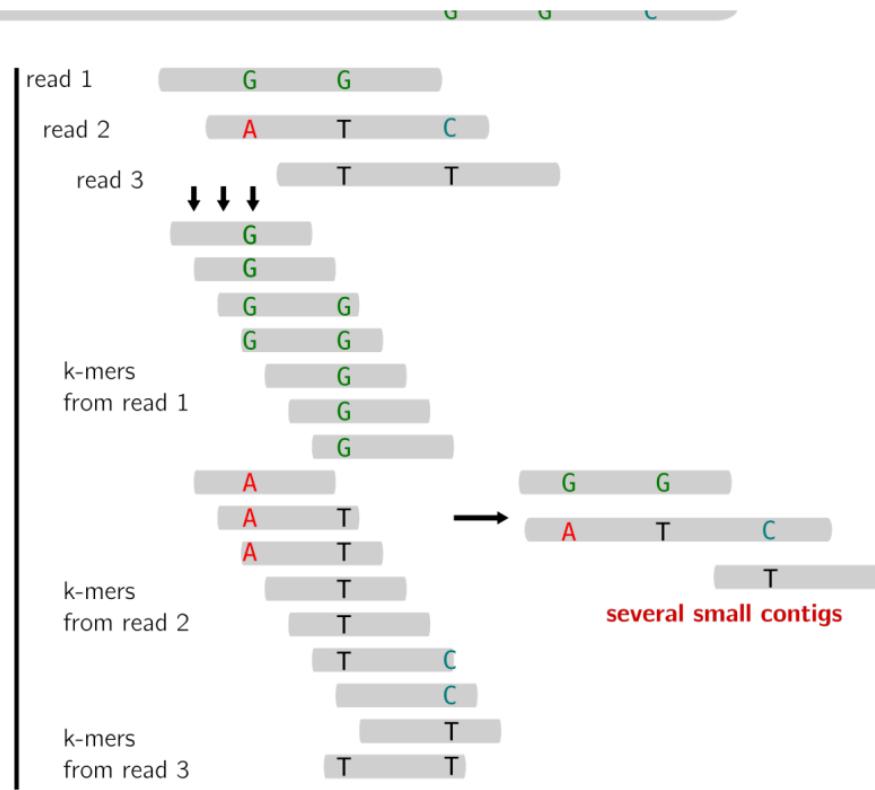
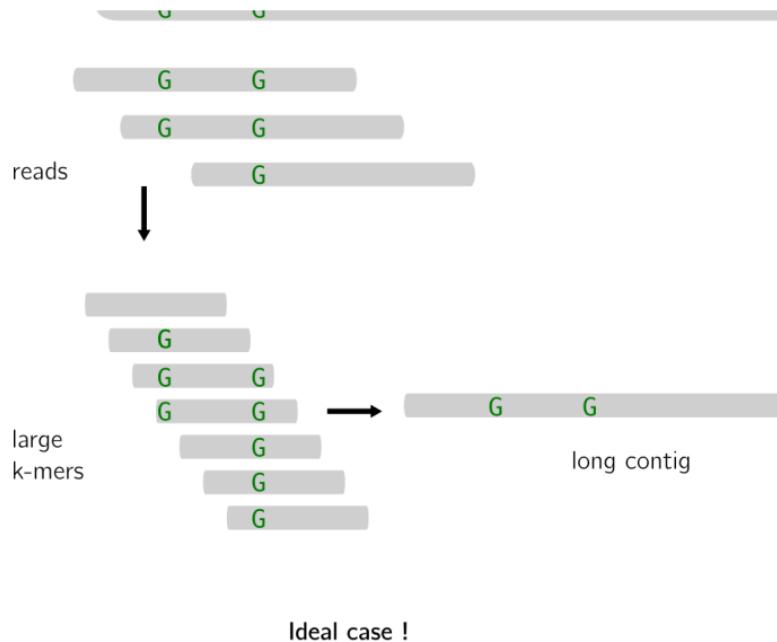
- de Bruijn graph or overlap graph?



- Sequencing errors and  $k$ -mers

Issue with large  $k$ -mers

## • Sequencing errors and $k$ -mers (ii)



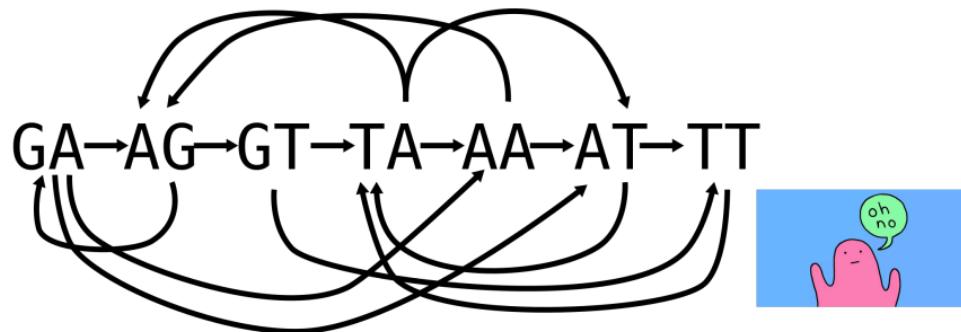
- **Sequencing errors and  $k$ -mers (iii)**

- large  $k$ -mers won't connect
- many spurious  $k$ -mers

- Sequencing errors and  $k$ -mers

Let's try small  $k$ -mers

read 1 GAGTAGAAATGAG  
read 2 AATAGAAATTAGT

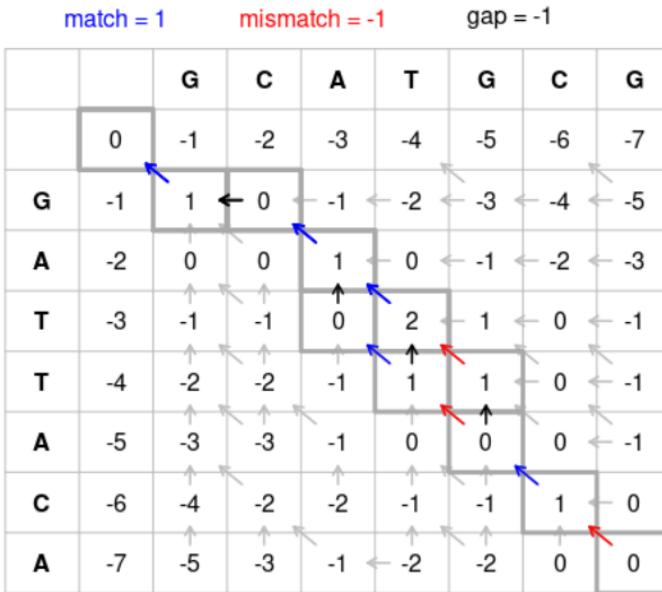


- the graph becomes too complicated: everything is a “repeat”
- we lose the advantage of having **long** reads

→ **de Bruijn graph out!**

- Overlap graph: inexact matches

GATTACA  
↔  
GCATGCG  
compute overlap?



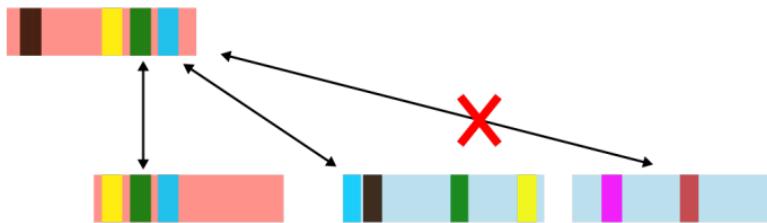
- Overlap graph: inexact matches (ii)

Quadratic alignment for each pair of reads

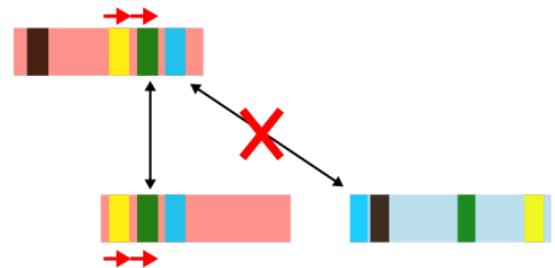
Quadratic number of comparisons to perform ...

- Overlap graph: drop alignment

1. find common seeds

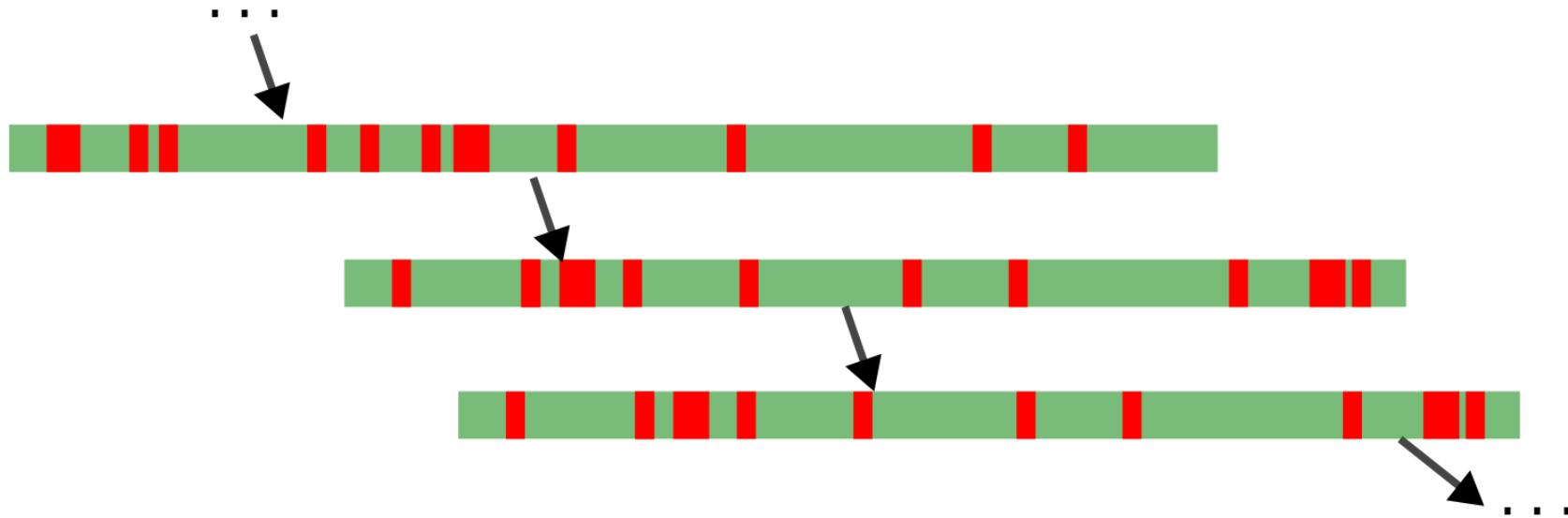


2. find if long chains of common seeds are in same order



Procedure called *anchor chaining*.

- How to get accurate contigs from noisy reads?



- Using coverage to remove noise: consensus

Reads:

The figure shows 12 DNA sequences (reads) aligned vertically. Yellow boxes highlight matching bases across the reads. The consensus sequence is derived from these aligned reads.

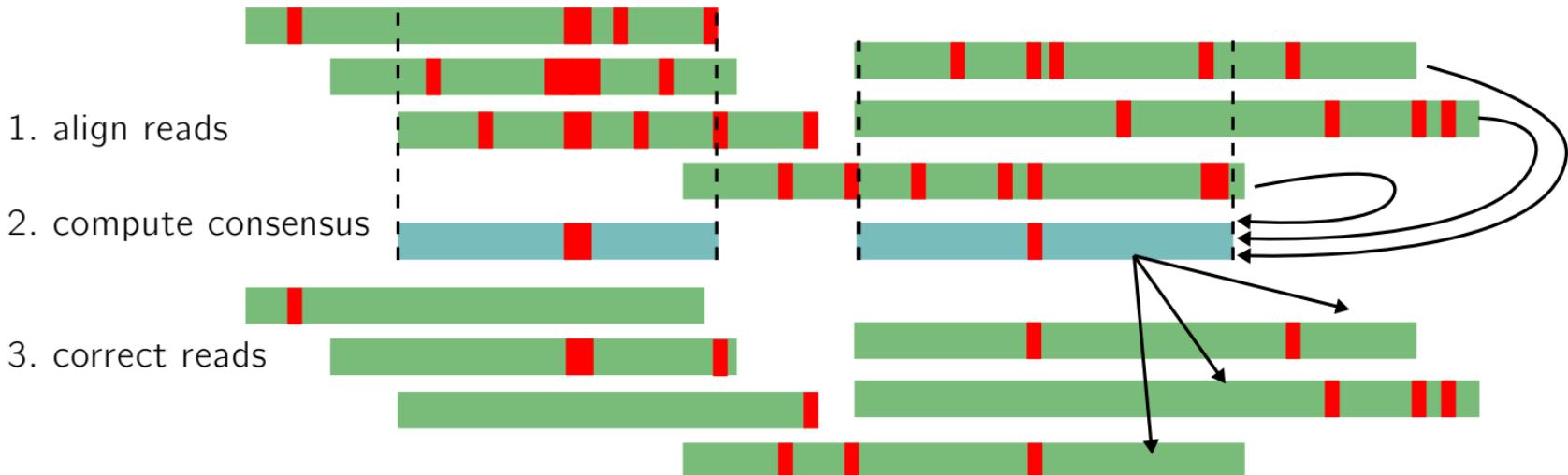
```

AAAGAAAGCACTGAATCA|TGGGACTT|CGAG
GAAAGCTCTCAACCAACGGACTGCGACTTT
ACCTCTCAAGCAACGGACTGCGACAAAAAG
TCTGAATCACCGACTGCGTCAAAAAGTGC
GAATCACCGACTGCGACAGTTGTGGTGG
TCAACCGCACTGCGACAATAAGTCCTGGTAT
ACGGACTGCGACAAAAAGTGTGGTATCCA
GACTGCCACAAAAAGTGGTGGTATCCAG
TGCGACAAAAAGTGGGTATCCAGAAT
GACAATAAGGGGGGTATCCAAAATTG
AAAAAGGGGTGGTATCCAGAATTTC
TAAGTGGGGTATCCAAAATTTCAGTT
    
```

Consensus:

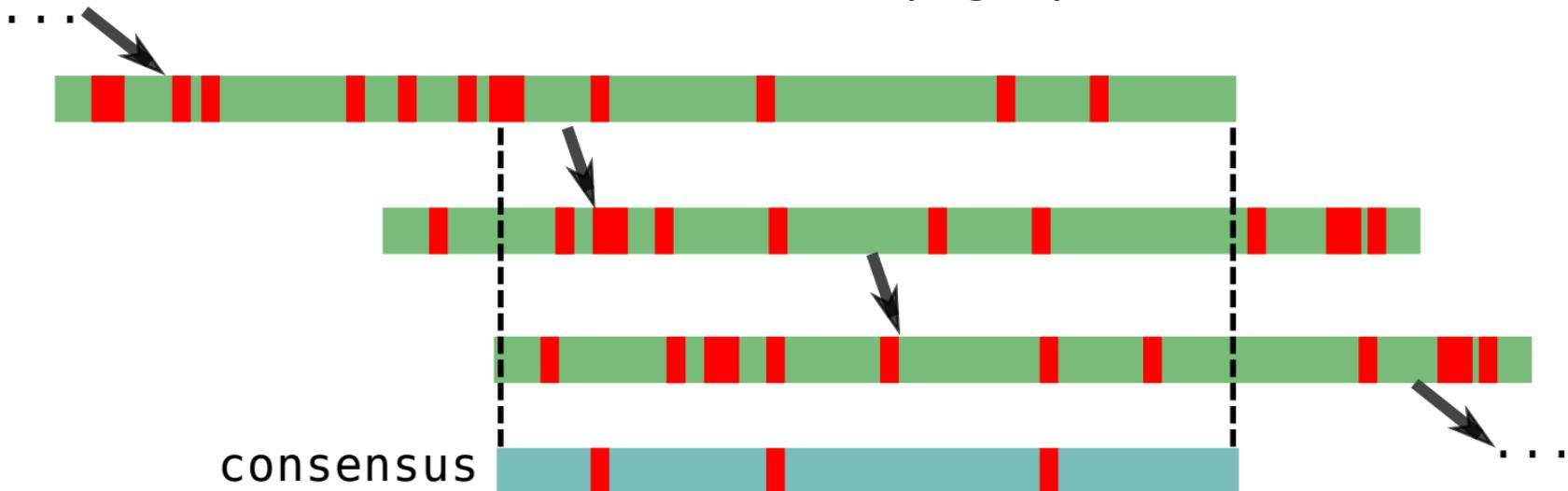
AAAGATAGCTCTGAATCAACGGACTGCGACAAAAAGTGGTGGTATCCAGAATTTCAGTT  
 1/1            4/7            9/10            6/11            3/4

- Consensus before assembly: correction

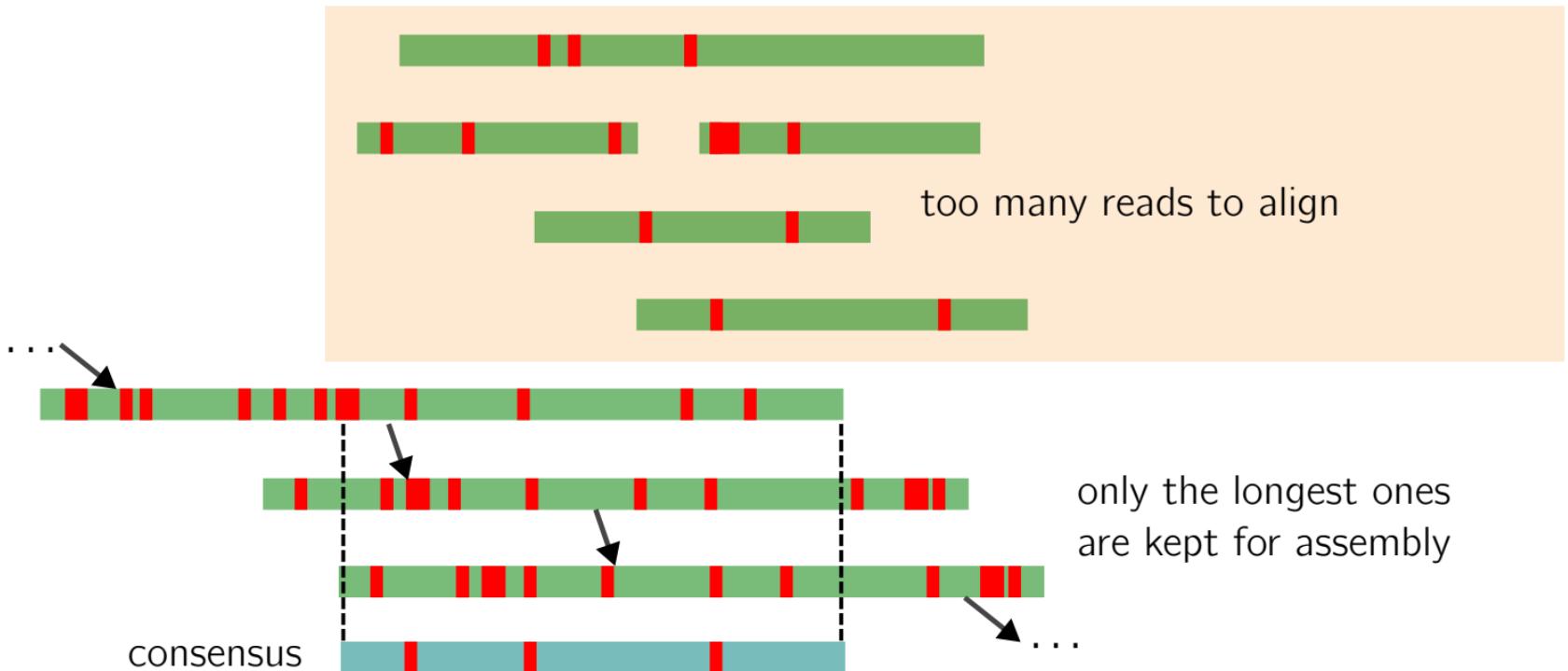


- Consensus during assembly (hence the OLC)

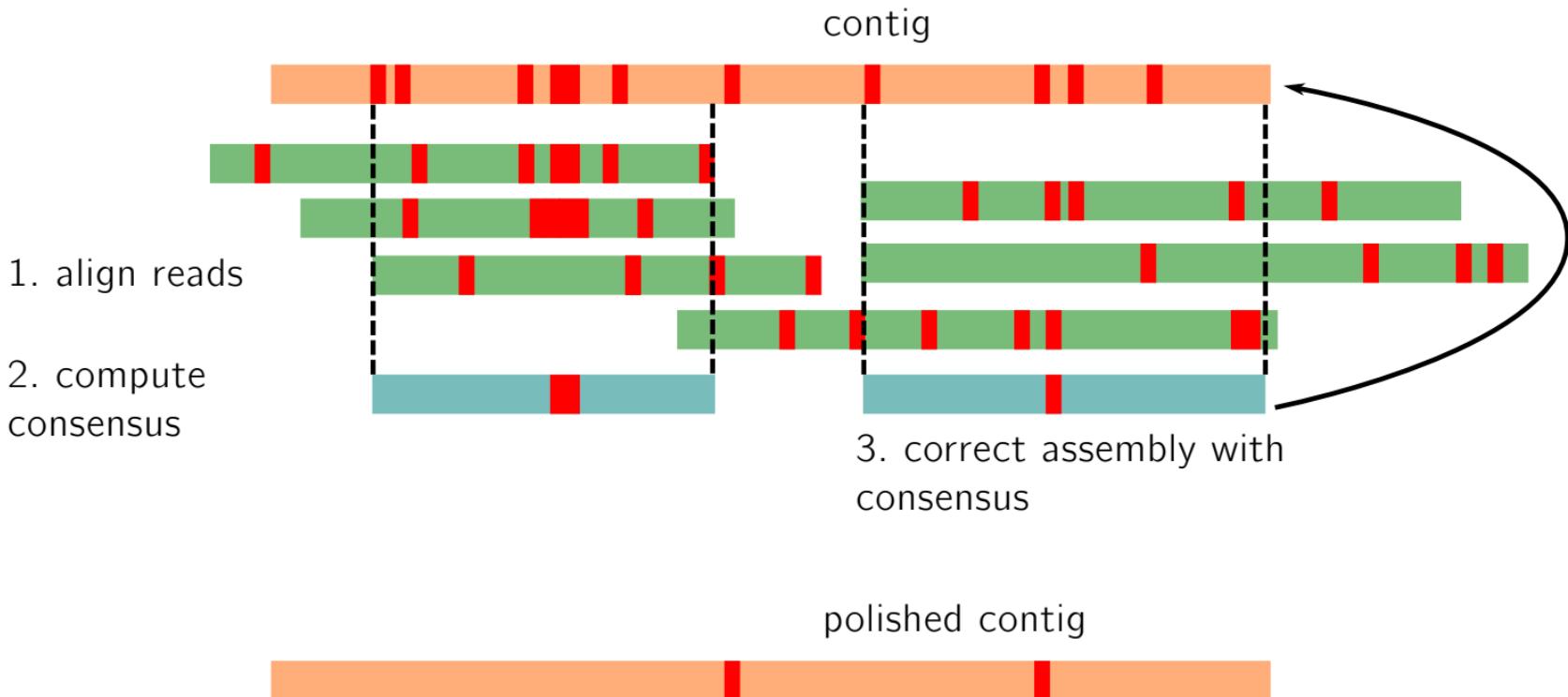
overlap graph



- Consensus during assembly. Yes but:



- Consensus after assembly: polishing



## • Correction/Consensus during assembly/Polishing

- Correction *x*
- Redundancy: 100X depth → 100X more bases to correct
- Consensus during assembly approx
- Do not use all reads
- Polishing ✓
- Correct each base of the genome once
- Use all reads

- Consensus destroys heterozygosity

reads

AATTGATCCGATACCC-GTAA-A  
AATTGGCGATACCC-GTAA-AG  
- ATTGATCCGA-ACCCCGTAA-A  
AATTGATCCGATACCC-GTAA-A  
    GCTCCGAGACCA-GTCA-ATTG  
    GCTCC-AGACCA-GTCA-ATTT  
        CCGAGACCA-GTCG-ATTGCAAA-  
        CCGAGACCA-GT-A-ATTGC~~G~~AAC  
        CCGACACCA-GT~~G~~AAATTGCAAAC

---

consensus AATTGATCCGAGACCA-GTCA-ATTGCAAAC

→ a mix between the two alleles



- Consensus destroys heterozygosity (ii)

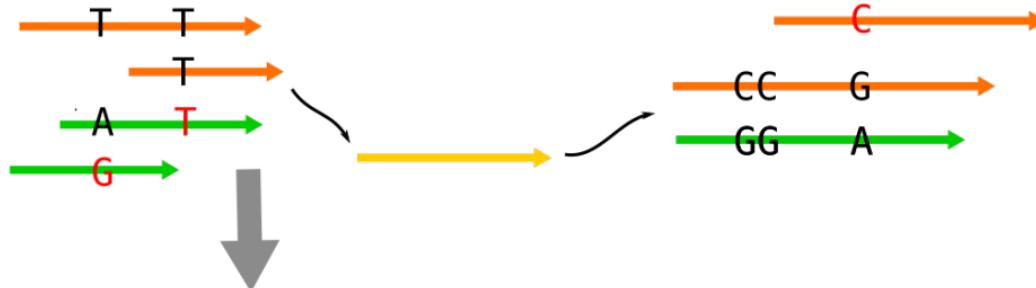
- Consensus destroys heterozygosity

♀ GGATGAAACTGCCGGTCAGGTACCCCCCTCTGAGCCGCCAAAATGTGCTGCCGGAC

♂ GGATGAAACAGGCCGGTCAGGAACACCCCCTCTGAGCCGGAAAATGTGCTGACCGGAC



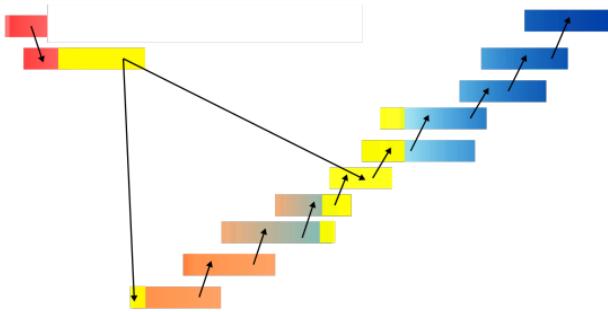
overlap graph+consensus



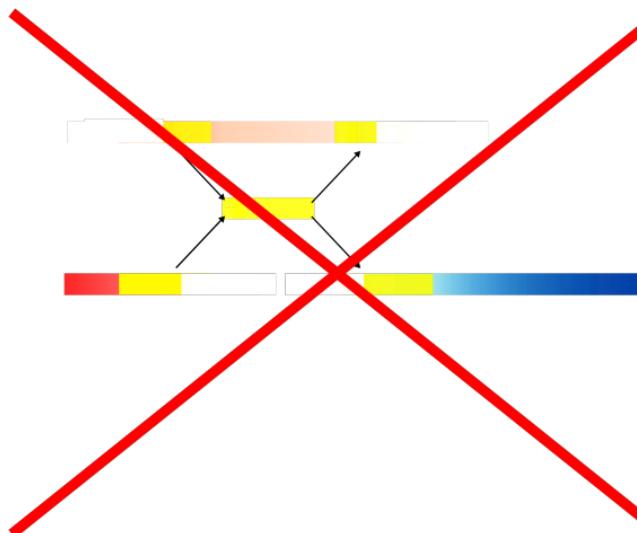
- **Consensus destroys heterozygosity (ii)**

Assembly concession: “haploid” assembly due to errors

- Method checkpoint: de Bruijn graph vs overlap graph



can deal with approximate overlaps  
(seed and chain)  
polishing



too many errors prohibit connectivity

- Data checkpoint: results for the *noisy super long boy*



100kb region from the genome



1 million reads → 100 contigs



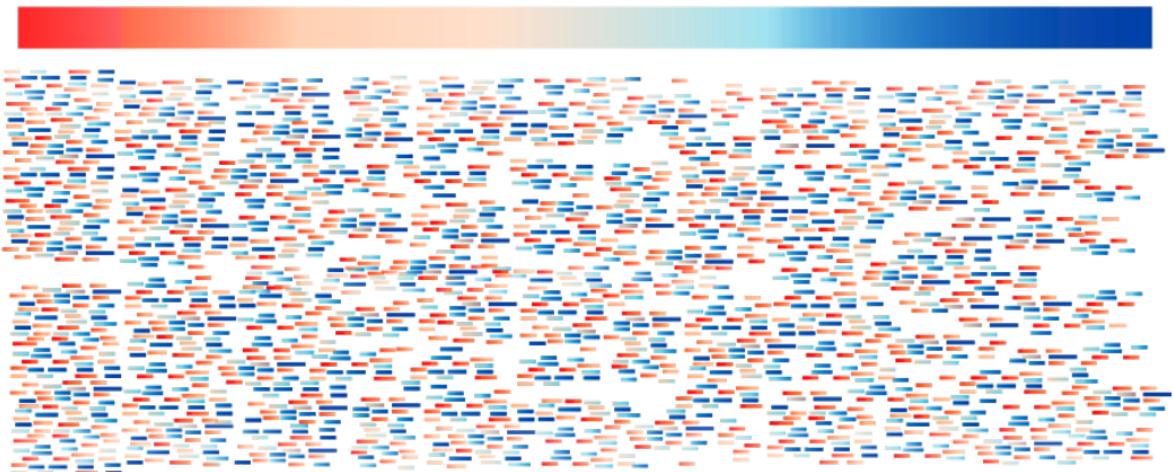
- Contigs can reach the chromosome's order of magnitude in length (megabases)
- Breaks due to very large repeats
- Contigs are chimeras of haplotypes

- Data checkpoint: results for the *noisy super long boy* (ii)

- Third experiment: *short boy's genome*

↳ [some region from the genome](#)

(only for the record, we actually don't have it)

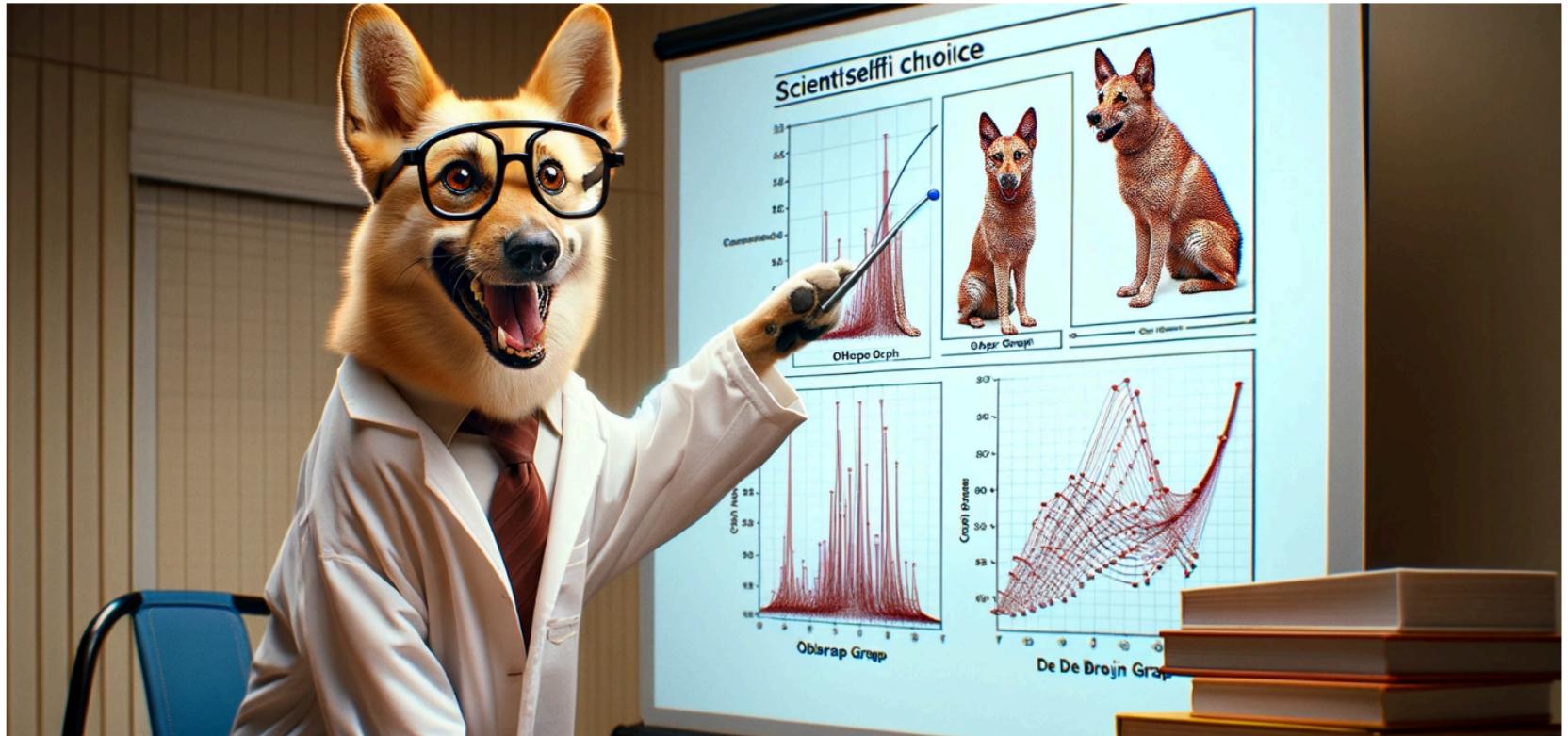


Genome size  
1 billion bases

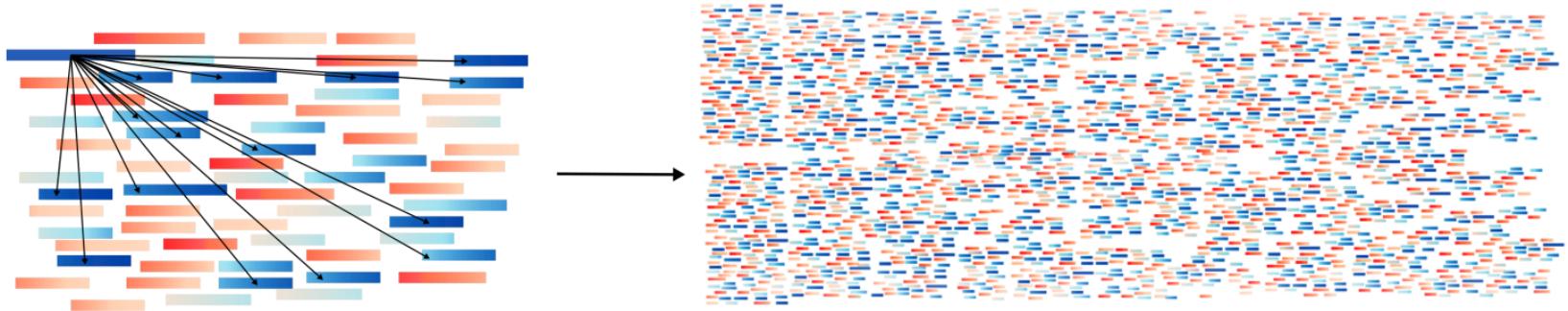
Reads  
1 billion  
size 100 bases

$\sim 10^9$  ...

# de Bruijn graph or overlap graph?



- Scalability issue for the overlap graph



At equal coverage we got:

1000  $x$  more reads  $\rightarrow$  1 million  $x$  more overlaps to check!

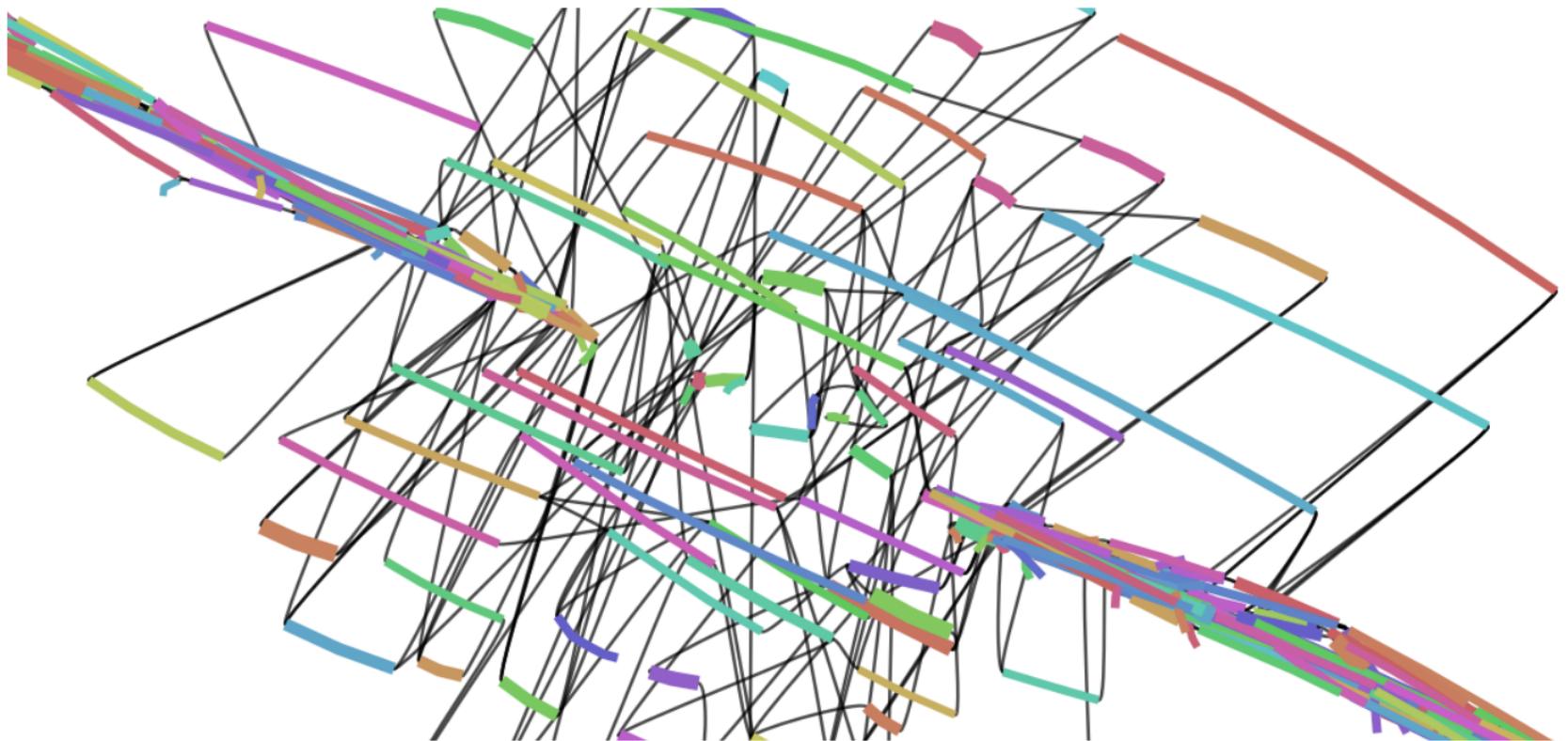
Overlap graph hardly scales to such a large number of reads/overlaps

→ Overlap graph out!

- de Bruijn graph on a real dataset



- de Bruijn graph on a real dataset ZOOMED IN



- Erroneous  $k$ -mers vs genomic  $k$ -mers

TAAGAAAGCTCTGAATCAACGGACTGCGACA

Reads:

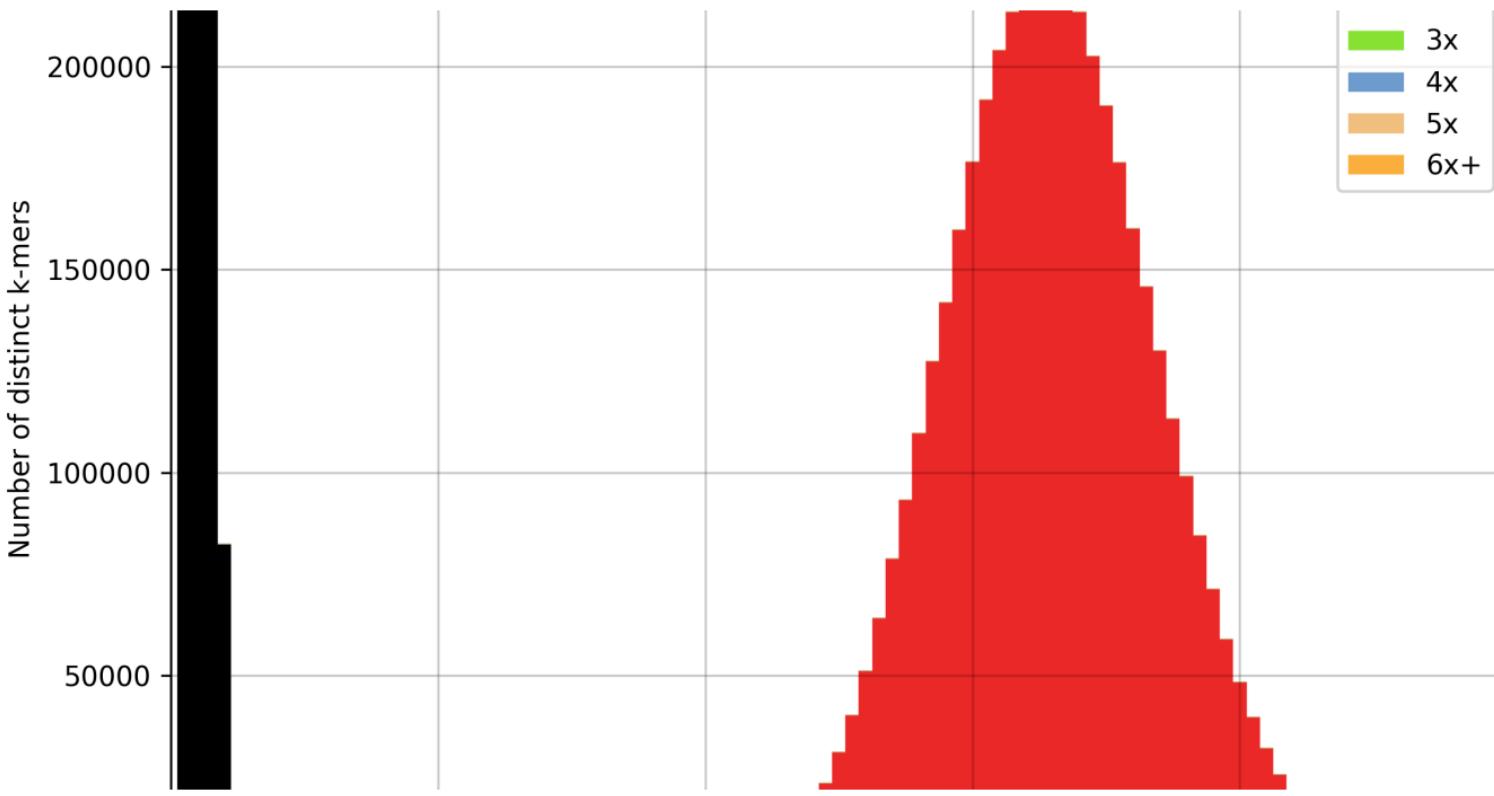
TAAGAAAGCTCTGAATCA  
AAGAAAGCTCTAAATCAAC  
AGAAAGCTCTGAATCAACG  
GAAAGCTCTGAATCAACGGA  
AAAGCTCTGAATCAACGGAC  
AAGCTCTGAATCAACGGACT  
AGCTCTGAATCAACGGACTG  
GCTCTGAATCAACGGCTGC  
CTCTGAATCAACGGACTGCG  
TCTGAATCAACGGACTGCGA

9 times	TCTGAAT
1 time	TCTAAAT
6 times	CAACGGA
1 time	CAACGGT

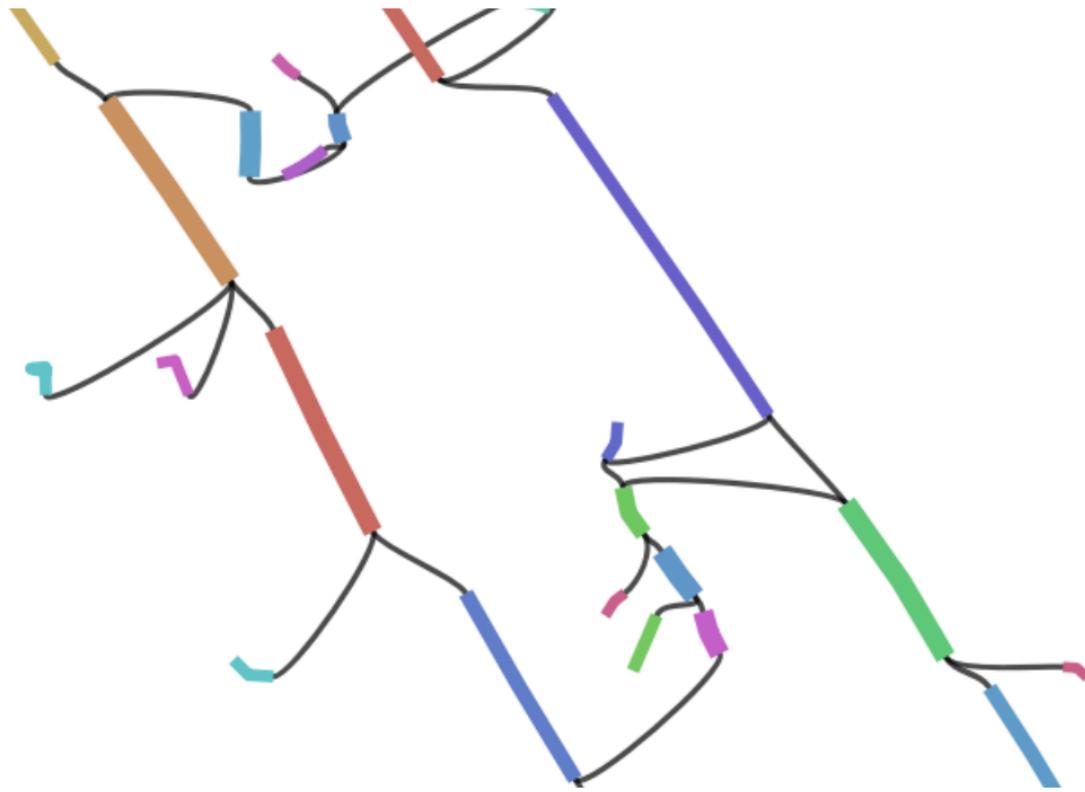
- **Erroneous  $k$ -mers vs genomic  $k$ -mers (ii)**

Erroneous  $k$ -mers are seen less than genomic ones

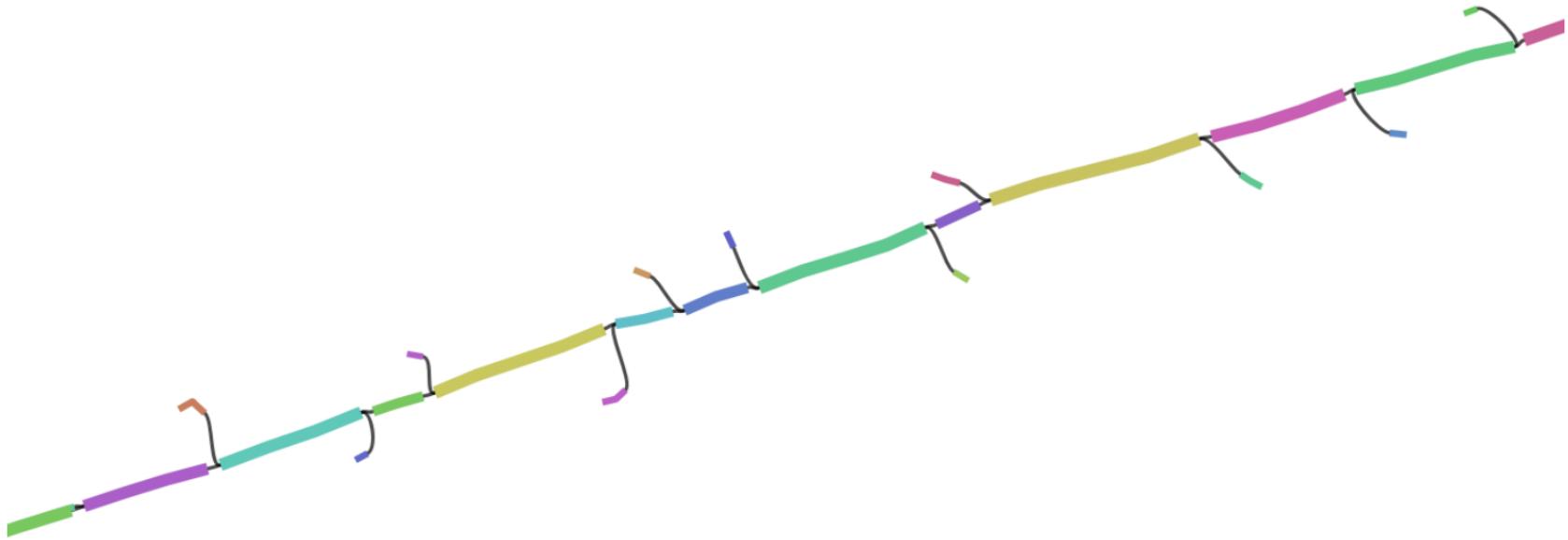
- **K-mer histogram**



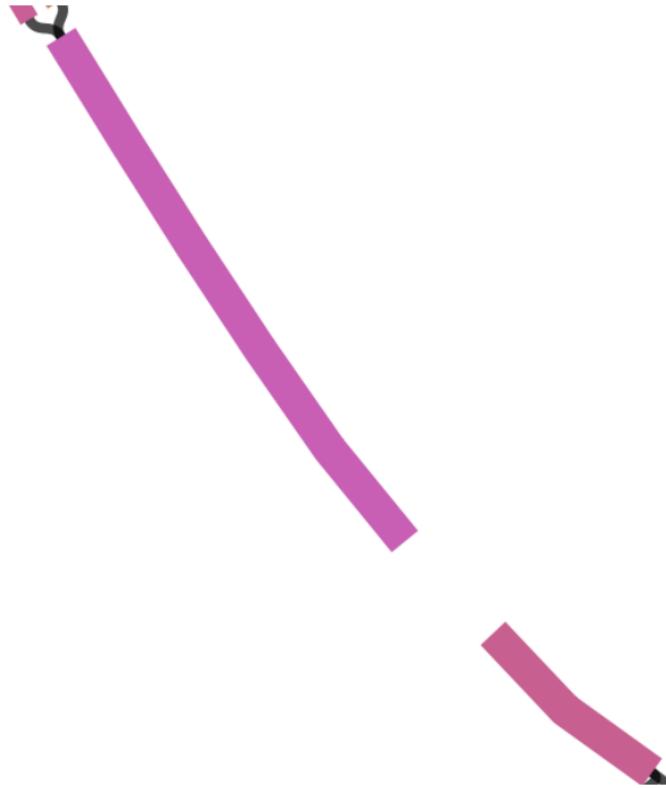
- Removing unique  $k$ -mers



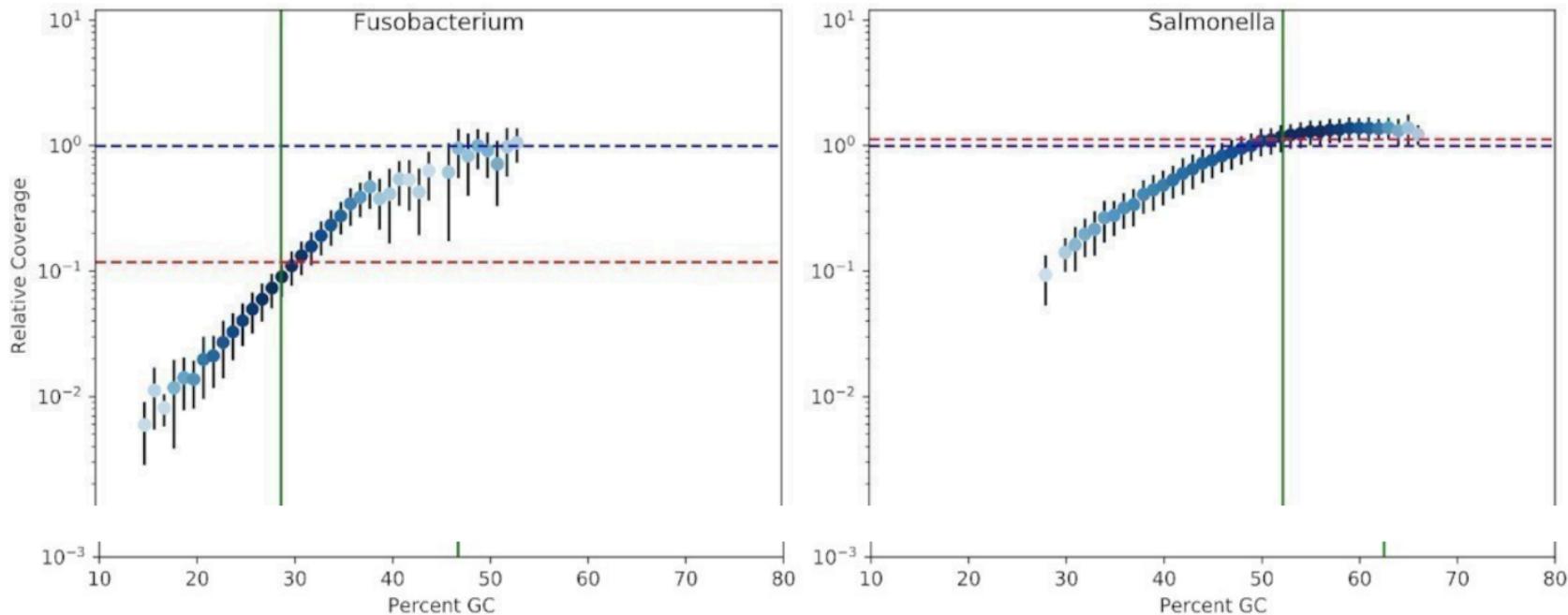
- Removing  $k$ -mers seen less than 3 times



- Removing  $k$ -mers seen less than 4 times



## • GC bias



Low GC region can be way less sequenced

- Errors in de Bruijn graphs

... TACAGGACTTACTGA... genome

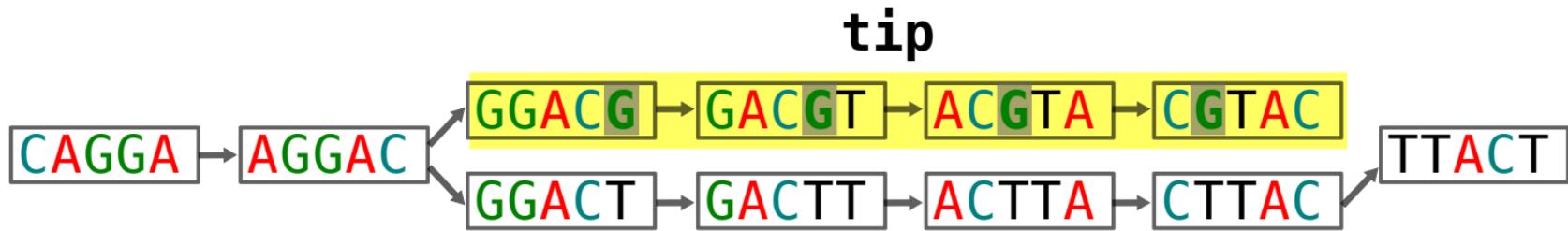
reads CAGGACTTA  
AGGACGTAC ← sequencing error  
AGGACTTAC  
GGACTTACT



- Errors in de Bruijn graphs

... TACAGGACTTACTGA... genome

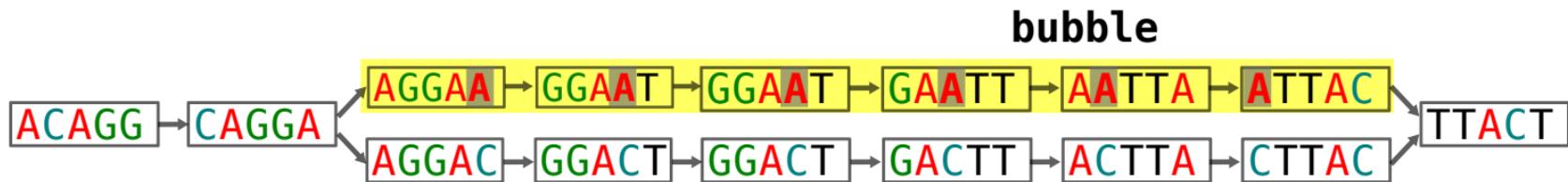
reads CAGGACTTA  
AGGACGTAC ← sequencing error  
AGGACTTAC  
GGACTTACT



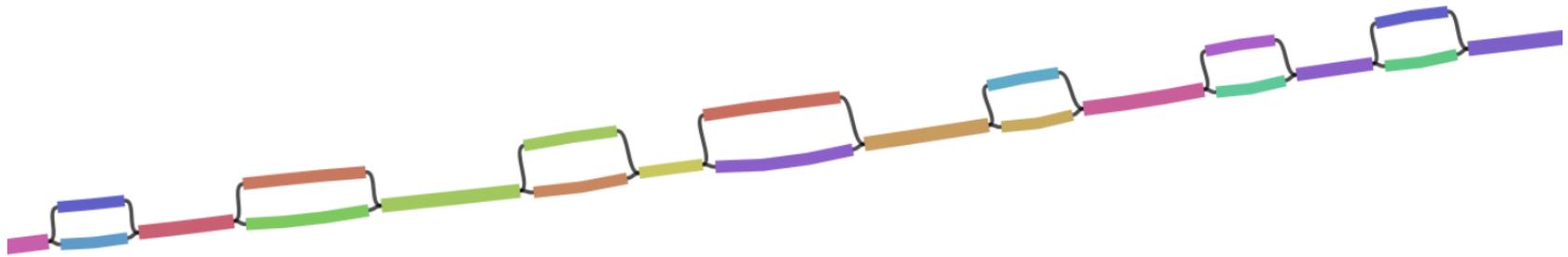
- Errors in de Bruijn graphs

... TACAGGACTTACTGA... genome

reads    ACAGGACTTA  
           CAGGAATTAC ← **sequencing error**  
           CAGGACTTAC  
           AGGACTTACT

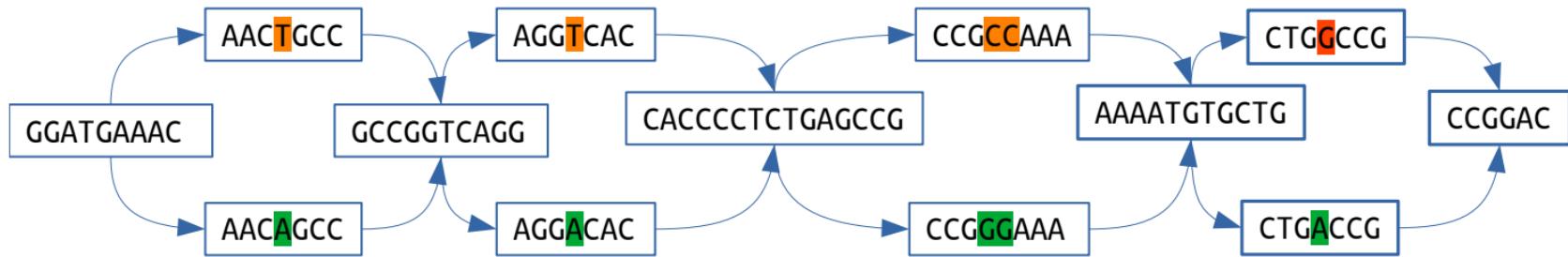


- de Bruijn graph on my diploid genome



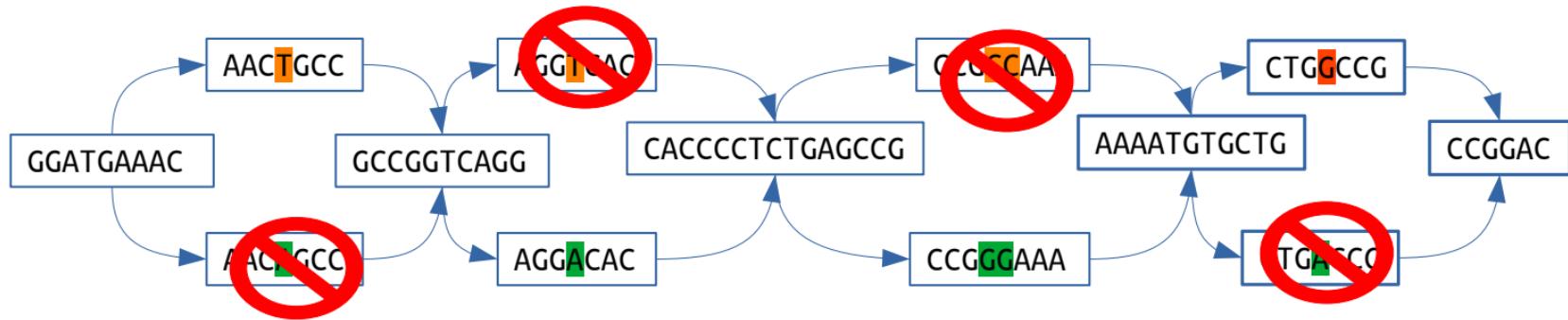
- Ploidy and de Bruijn graph

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC



- Bubble crushing

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC



Assembly:

- Variants are not “lost”

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC

Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGCCGGAC

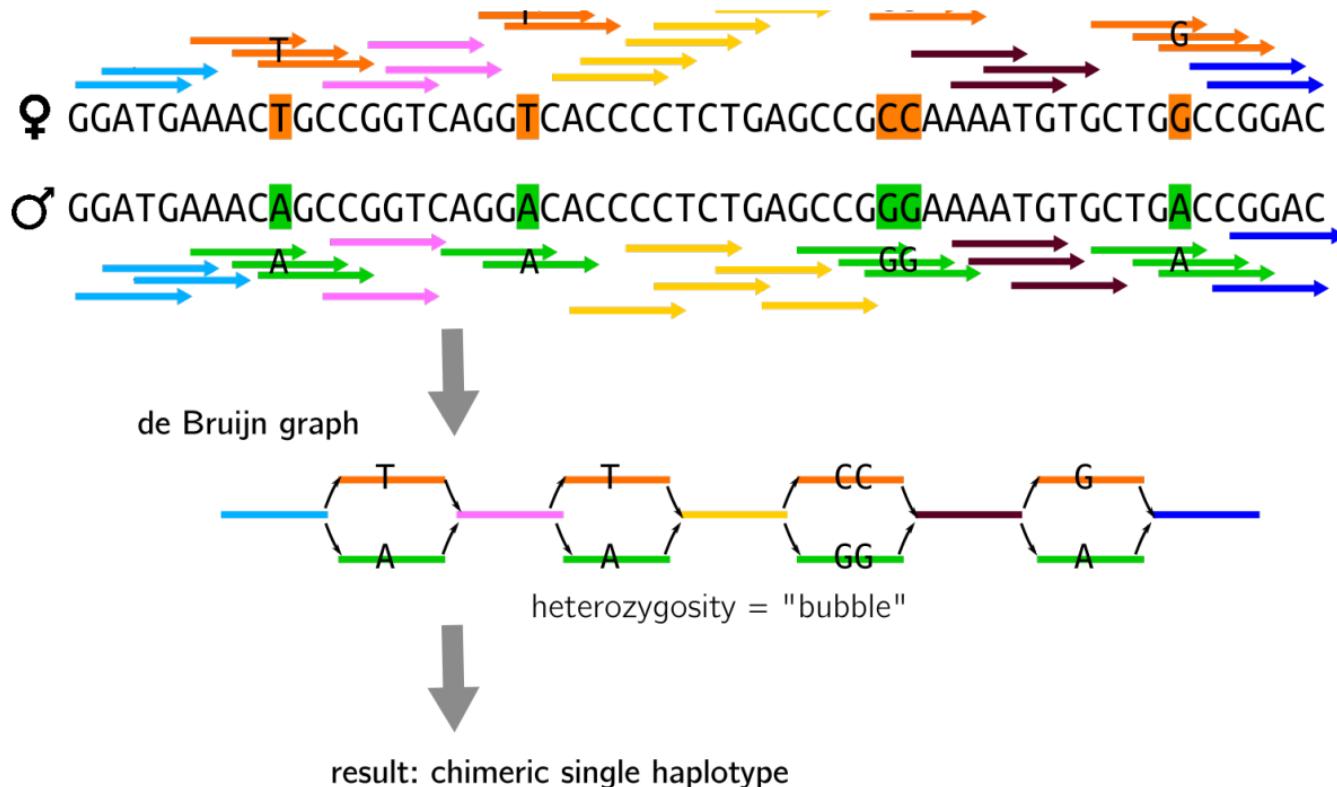
Reads:

GATGAAACTG  
ATGAAACAGC  
TGAAACAGCCG  
GAAACTGCCGG  
AAACTGCCGGT  
AACAGCCGGTC  
ACAGCCGGTCA  
CTCCCCCTCAAC

- **Variants are not “lost” (ii)**

We can align the reads to the assembly and do variant calling

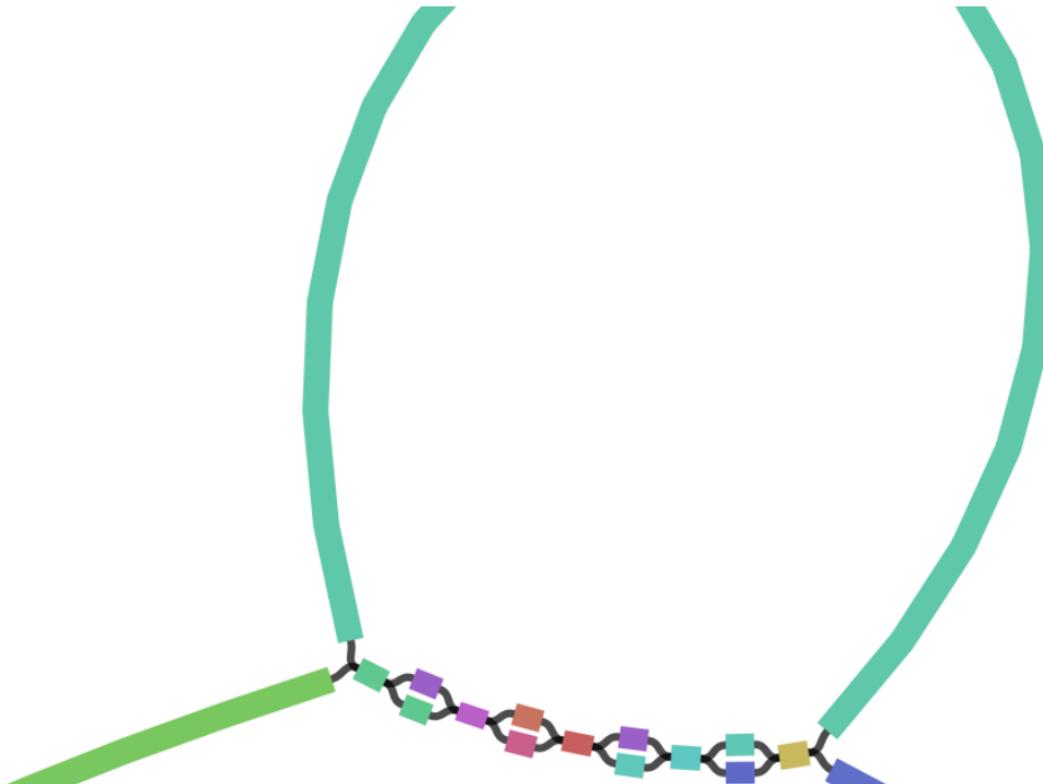
## • Haploid assembly



- Haploid assembly (ii)

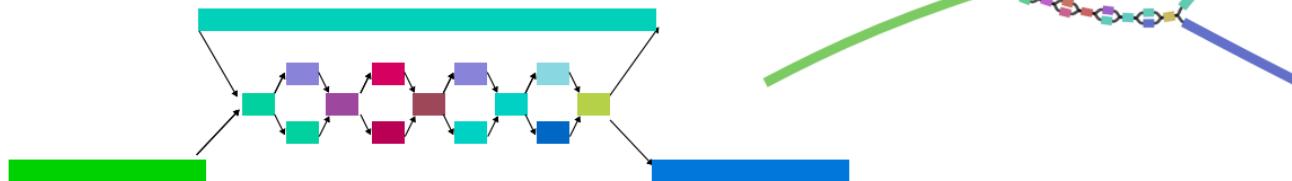
Assembly concession: haplotypes are collapsed when using short reads

- Paralog genes/repeats

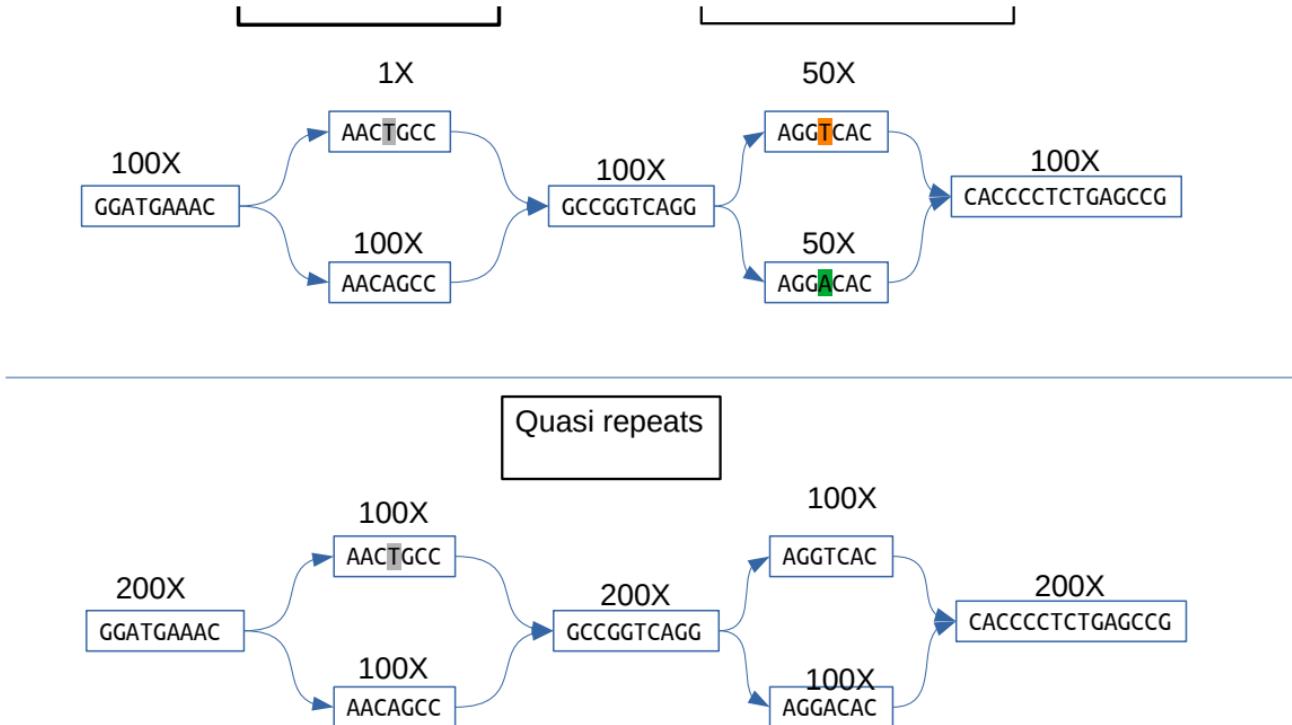


- Paralog genes/repeats

compacted de Bruijn graph



- Paralog genes/repeats in graphs

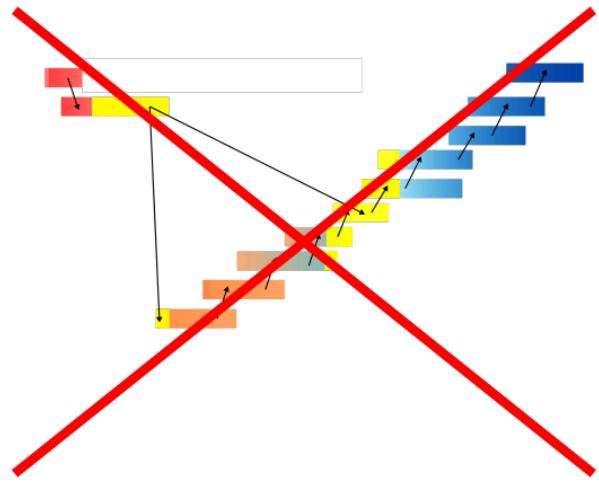


- Paralog genes/repeats in graphs (ii)

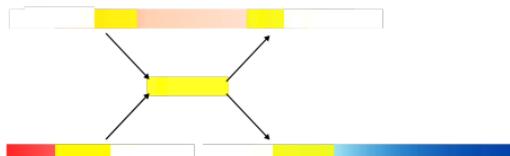
Treating erroneous bubbles is simple, all the others are complex

- Method checkpoint: de Bruijn graph versus overlap graph

Overlap graph



(compacted) de Bruijn graph



- Data checkpoint: *short boy* results



100kb region from the genome



1.000.000.000 reads → 100.000 contigs



- Very fragmented assembly of short contigs (mostly below 100kb)
- Very high base accuracy
- Contigs are chimeras of haplotypes
- Can miss low GC content

- Data checkpoint: *short boy* results (ii)

- Fourth experiment: *golden boy's genome*



- Fourth experiment: *golden boy's genome* (ii)

Billion \$ project → cancelled

- **(Time accurate) recap**

**Sanger**

body

**Illumina**

body

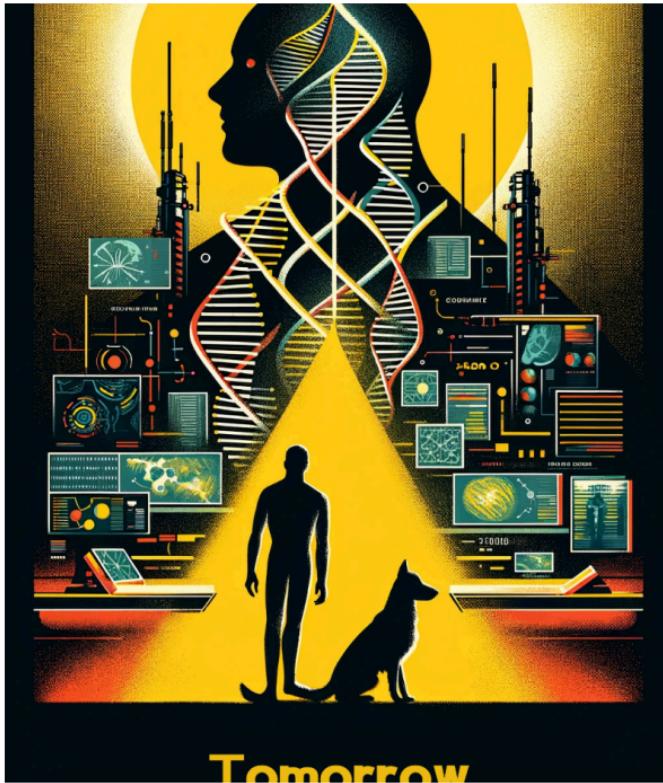
**Long reads: Oxford Nanopore or PacBio**

body

**HiFi**

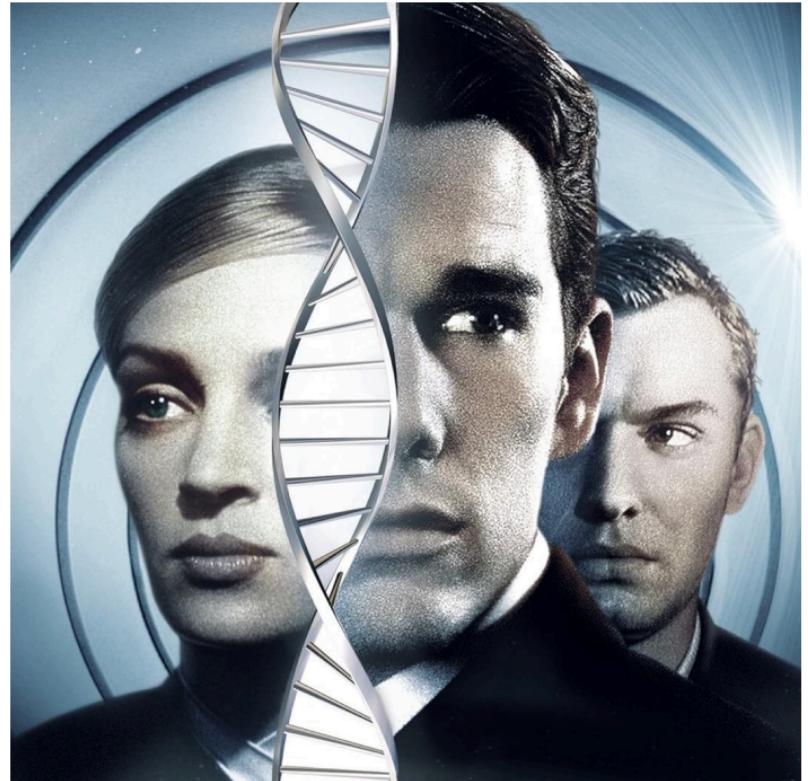
body

- Back to the present



## • Challenge 1: Scalability

- Human Genome project (2001)
- 1000 Genomes project (2015)
- 10k Genomes project (2016)
- 100k Genomes project (2018)
- 500K UK genomes (2023)



- **Challenge 1: Scalability (ii)**

Many ambitious sequencing projects beyond human: Earth biogenome project, Vertebrate genome project ...

## • History

How long to assemble a human genome?

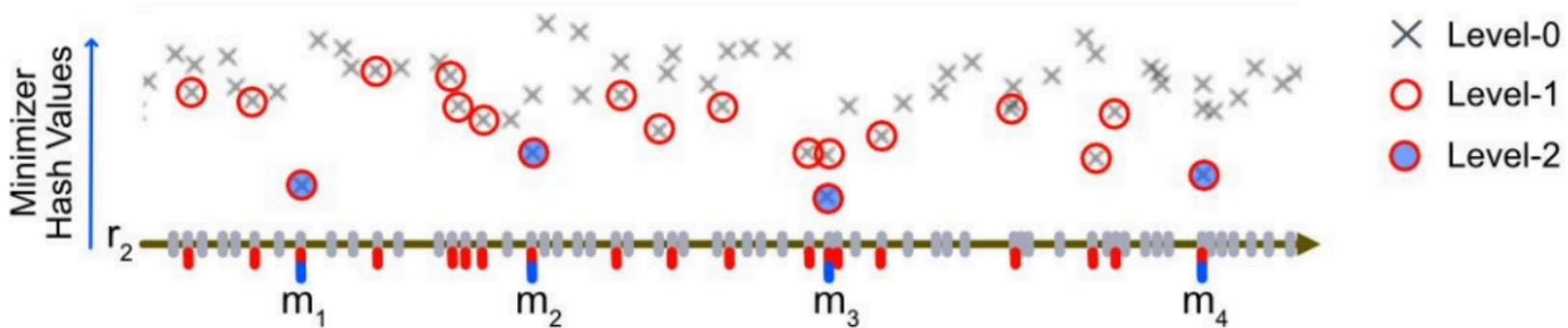
- Sanger: **MANY CPU years**
- Illumina (Overlap graph): **2 CPU months**
- Illumina (De Bruijn graph): **A CPU day**
- Long reads (Alignment): **2 CPU years**
- Long reads (Anchors chaining): **20 CPU days**
- HiFi (Anchors chaining): **2 CPU days**
- HiFi (De Bruijn graph): **A CPU hour**

**Algorithms and data structures matter!**

Also long and precise reads are easier to assemble

- Very fast genome assembly with HiFi

Human genome assembled within 2 hours (Peregrine assembler) and 10 minutes (RMBG assembler)



- Telomere to telomere assembly?



- Telomere to telomere assembly? (ii)



## • Challenge 2: Telomere to telomere chromosomes

Main problems:

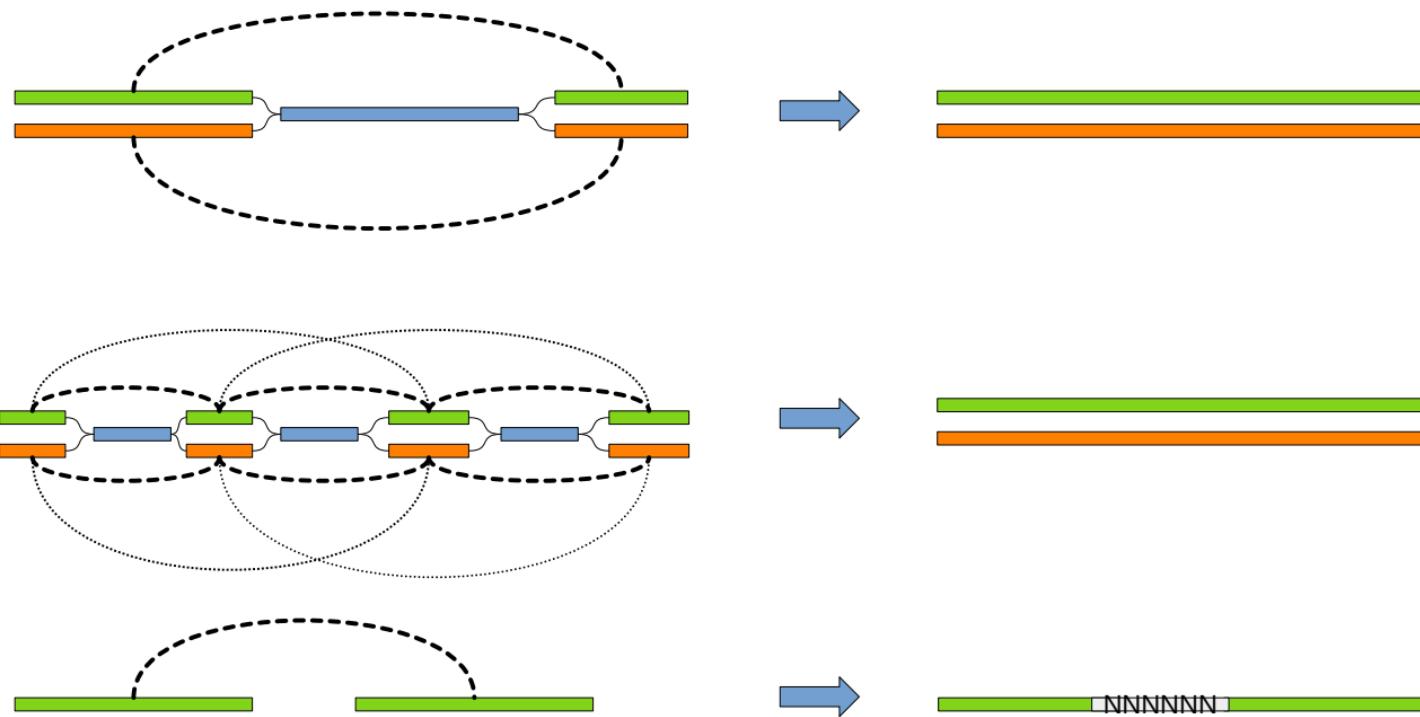
- Very large exact repeats
- Very similar sequences across the genome
- Low complexity regions
- Mosaic repeats

Need long distance information **AND** high base accuracy

- **Scaffolding**

Use long range information to order contigs into “scaffolds”

- Scaffolding (ii)



- Reference-based Scaffolding/assembly

Pros	Cons
body	body

- Map the reads on a reference and compute a consensus (Medaka)
- Use a reference assembly as existing contigs (SPAdes)
- Use one (or several) related references genomes to order contigs (Ragout2)

- **Telomere-to-Telomere consortium**

Has produced in 2022 a complete human genome with one contig per chromosome !

- 30x PacBio HiFi
- 120x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- Arima Genomics HiC
- BioNano DLS
- 100 authors from 50 labs

- Diploid assembly

(B)

Diploid outbred genome



Unphased maternal and paternal contigs



Haplotype-mosaic reference assembly



Assembled maternal and paternal chromosomes



- **Telomere-to-Telomere diploid human reference**

**T2T-YAO released in 2023 a complete human genome with one contig per chromosome !**

- 92x PacBio HiFi
- 336x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 584x Arima Genomics HiC
- BioNano DLS
- Illumina HiSeq 150bp for the son and parents (with 278x and 116x coverage, respectively).

- **The human genome is not THAT hard**

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)



- The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)
- Axolotl (32Gb)



- **The human genome is not THAT hard**

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)



- **The human genome is not THAT hard**

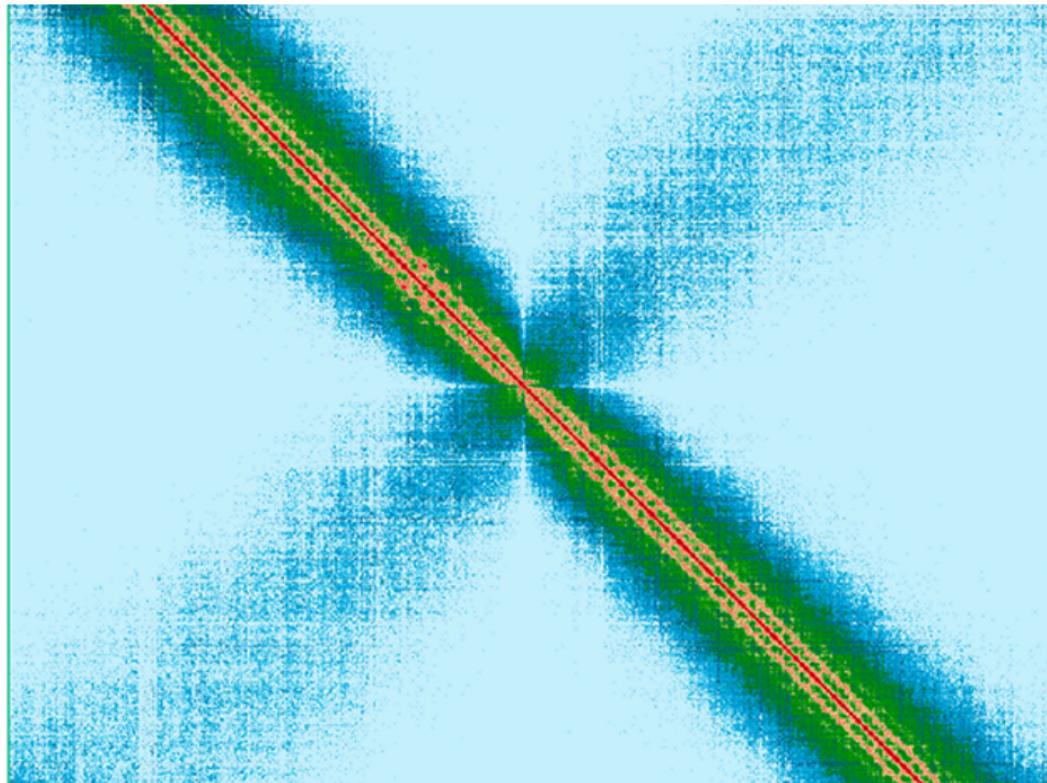
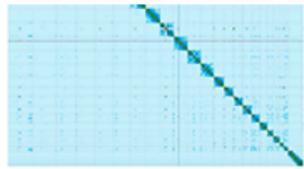
Hall of fame of largest assembled genomes of their time:

- The human genome is not THAT hard (ii)

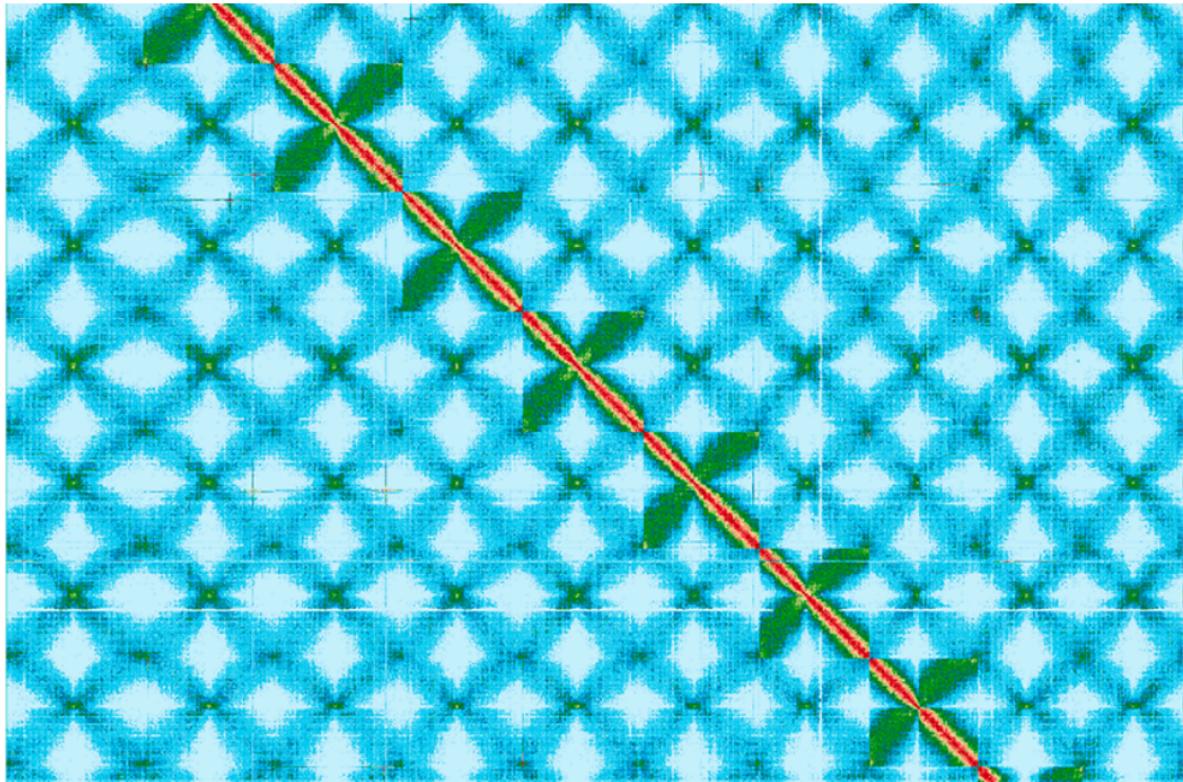
- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)
- Mistletoe (90Gb)
- Metagenomes ...



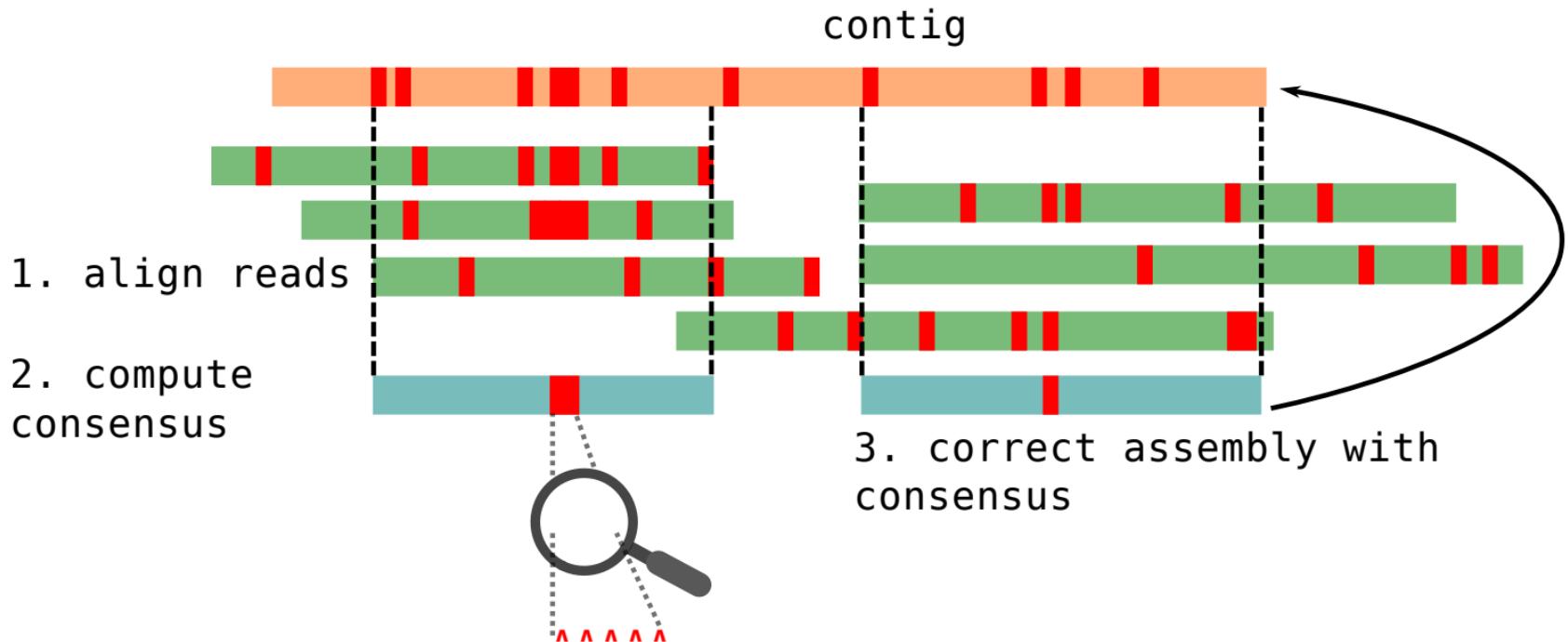
- The human genome seems small



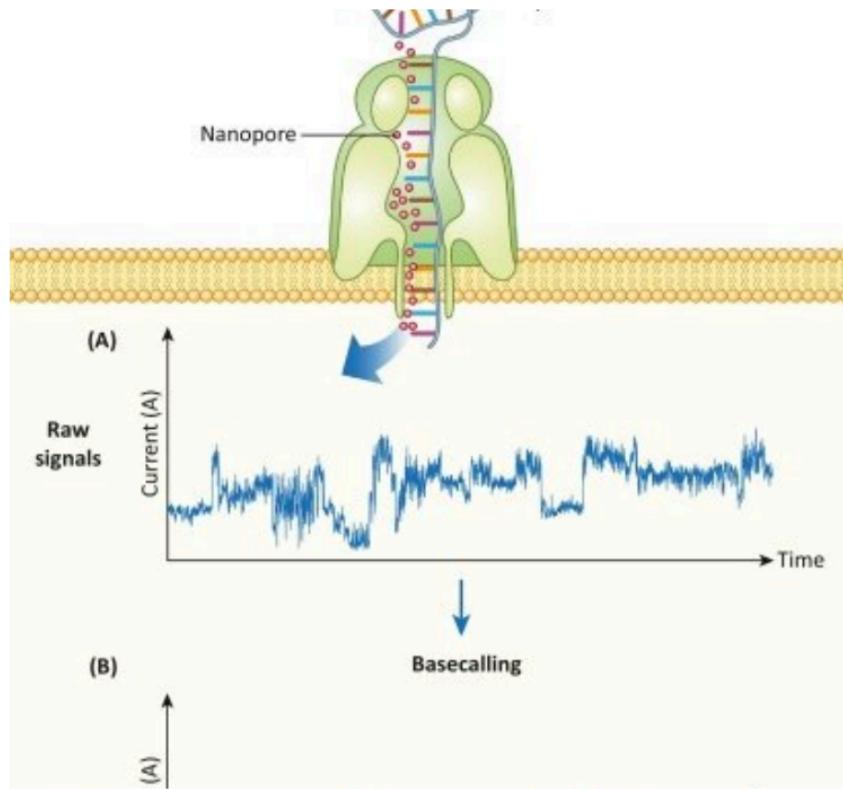
- The human genome seems really small



- Challenge 3: Base level accuracy



- Homopolymers are hard to read



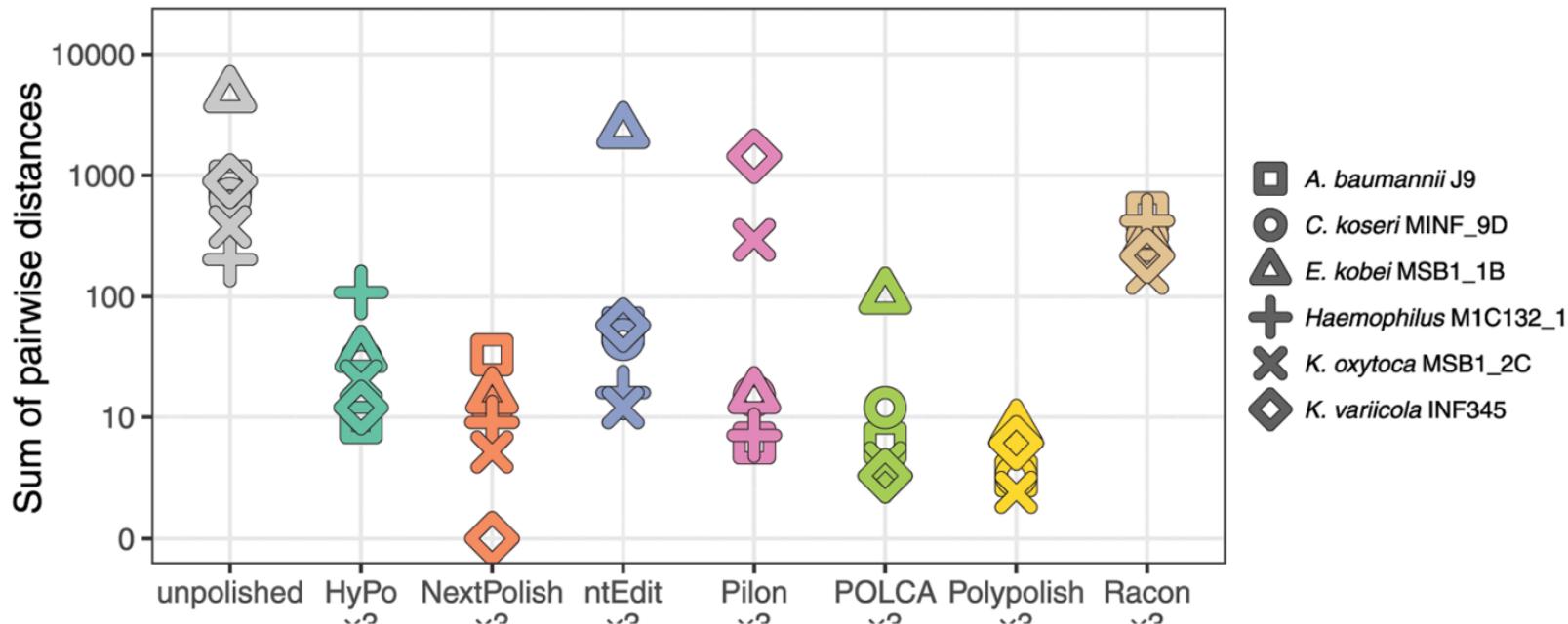
- **Systematic errors**

Polishing with Illumina data can improve the final error rate

- Systematic errors (ii)

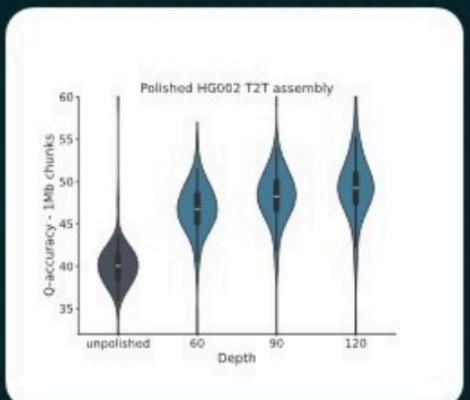
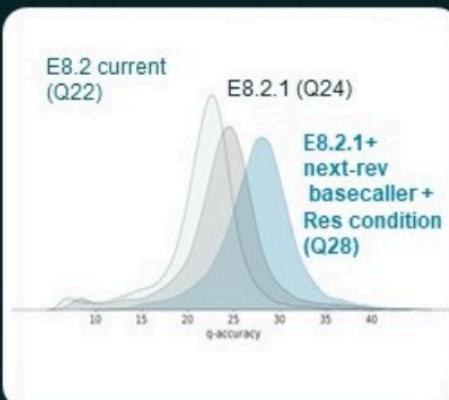
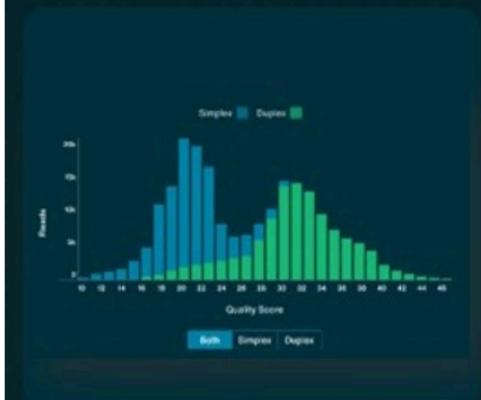
### A. Single-tool short-read polishing

ALE change: 0 110696 113366 87707 113056 113061 115623 82446  
 total distance: 7635 212 74 2519 1775 128 28 1867



- Basecalling progress: Dorado years

It's been an exciting 6 months in Nanopore R&D and Apps teams



Q20  
Simplex



Q28  
Simplex



Q50  
Nanopore Only  
Assemblies

## • Replication outside nanopore HQ

Post from Ryan Wick's bioinformatics blog (<https://rrwick.github.io/>) reports Q20 reads accuracy and Q60 assemblies on 9 bacterial assemblies

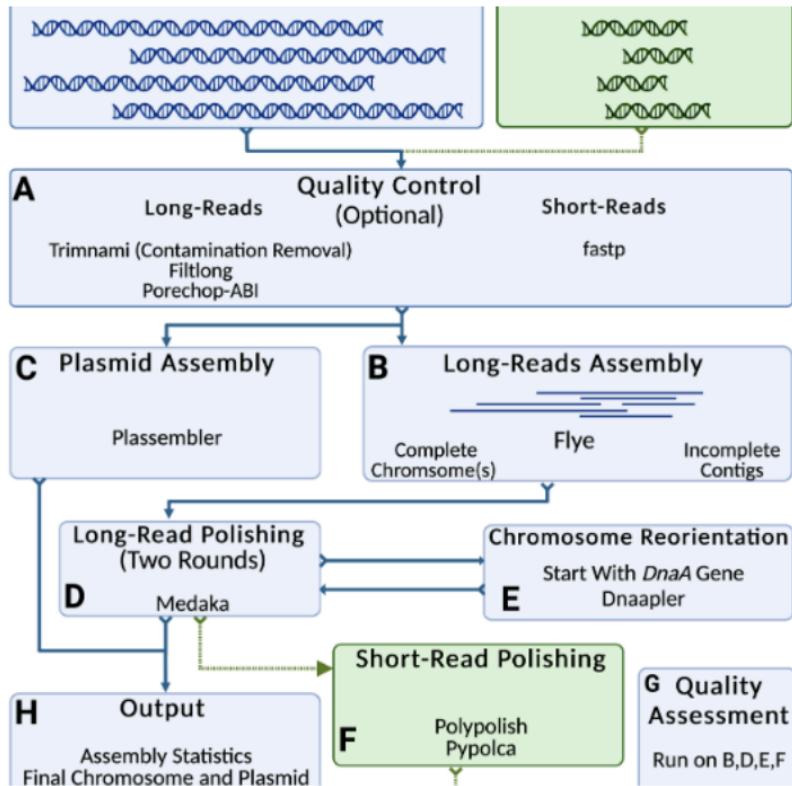
Average	Read accuracy	Assembly accuracy
mean	97.7%, Q16.4	4 errors, Q60.43
median	99.1%, Q20.5	2 errors, Q60.43
mode	99.4%, Q22.2	NA

## • HiFi-like Nanopore data ?

(Near) error-less very long reads we have several promising improvements ahead:

- Very fast assembly
- T2T chromosomes with less data
- Higher consensus accuracy
- Popolyplid assemblies

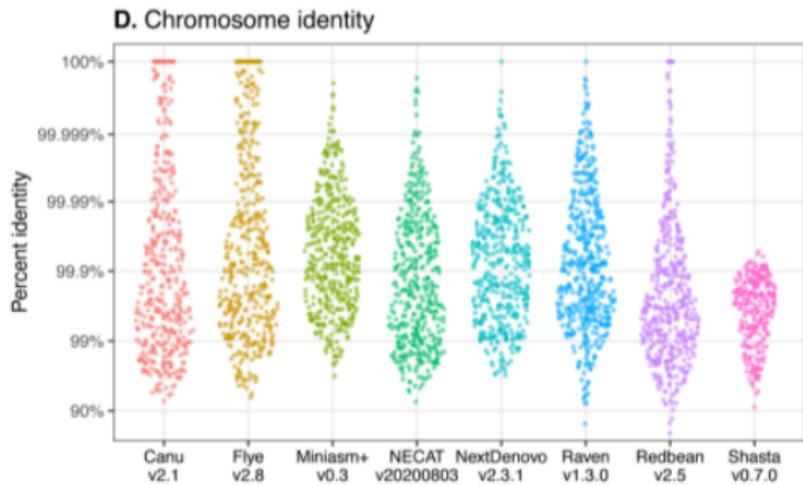
## • Challenge 4 : Assembly as a software



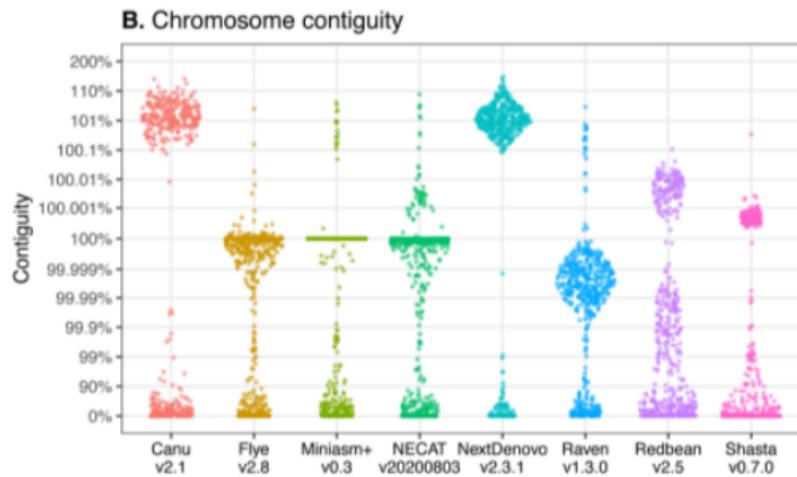
## • Challenge 4 : Assembly as a software (ii)

From Hybracter: Enabling Scalable, Automated, Complete and Accurate Bacterial Genome Assemblies

- Assemblers behave differently

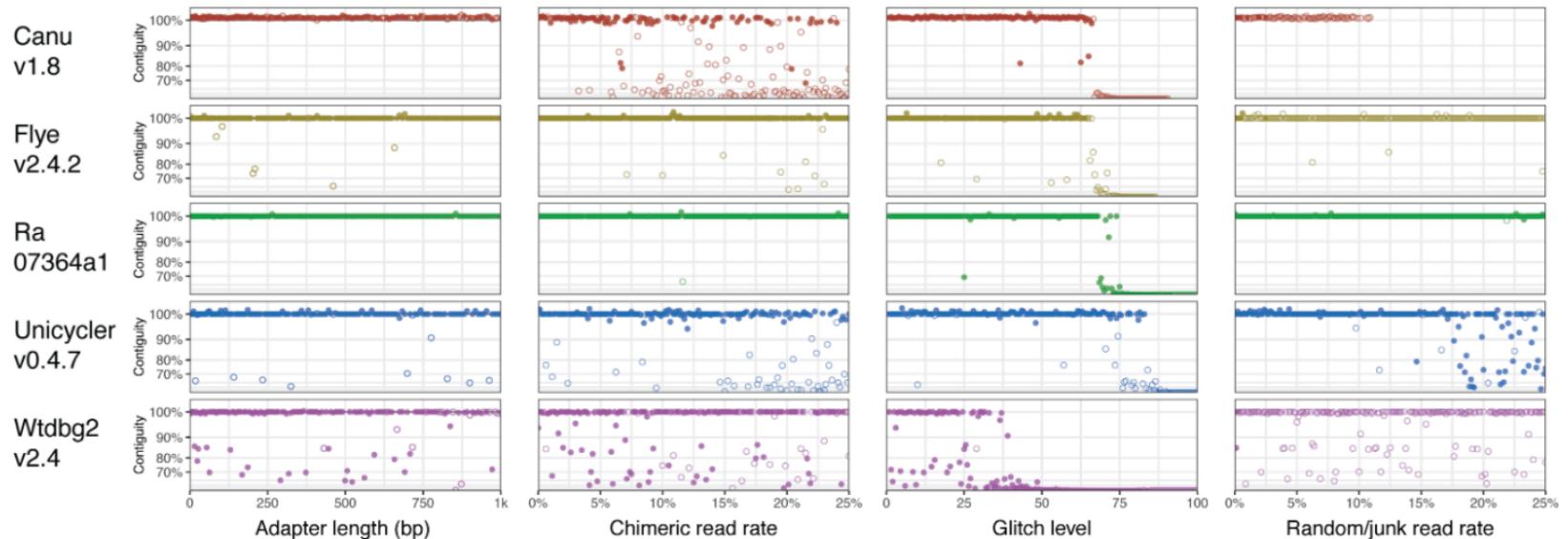


- Assemblers behave differently (ii)



From <https://github.com/rrwick/Long-read-assembler-comparison>

- Software robustness



From <https://github.com/rrwick/Long-read-assembler-comparison>

- **An assembly is a model**

1. Assemblies contain errors
2. Different tools can produce very similar assemblies
3. A single tool can produce very different assemblies with small changes of parameters(!)

# The (first) end

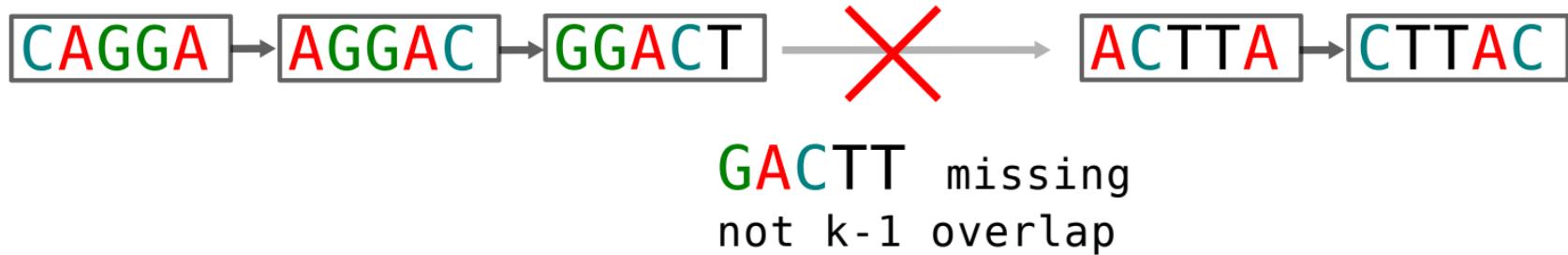


## Advanced points

If we have time, we'll review everything (while doing this course, I doubt it ...) Else, pick one:

- Multiple k assembly in de Bruijn graphs
- HiFi de Bruijn Assembly
- An overlap graph limitation with noisy long reads (and current fixes)
- The repeat graph

- Coming back to a de Bruijn graph limitation: fixed overlaps



GGACT and ACTTA overlap is only of size 3 !

- A too small  $k$  is not a solution
- We would like larger  $k$ 's but miss connections

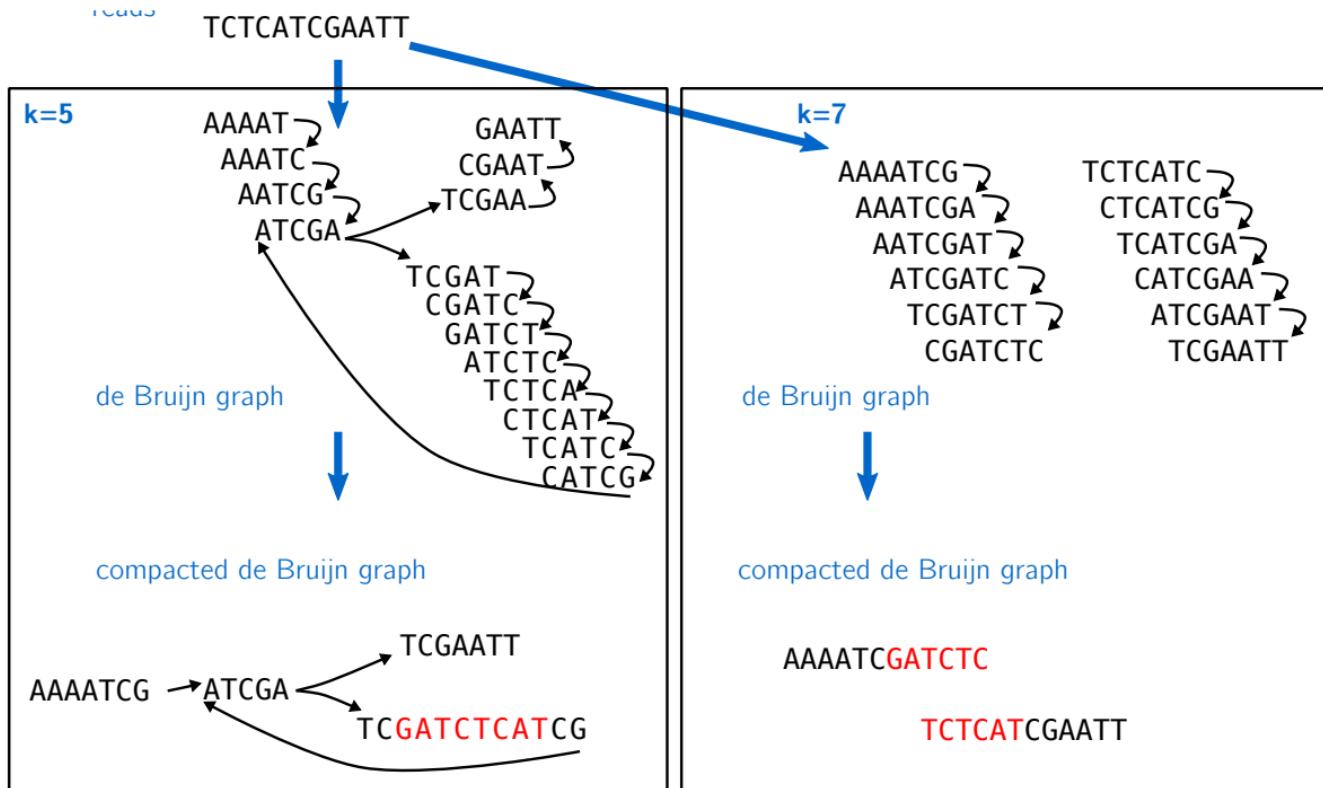
- **Multiple  $k$  assembly**

Most de Bruijn graph assemblers can now perform several assemblies with different  $k$ -mer sizes to produce an improved super assembly

**Exercise**

body

## • Multiple $k$ assembly



- **Multiple  $k$  assembly (ii)**

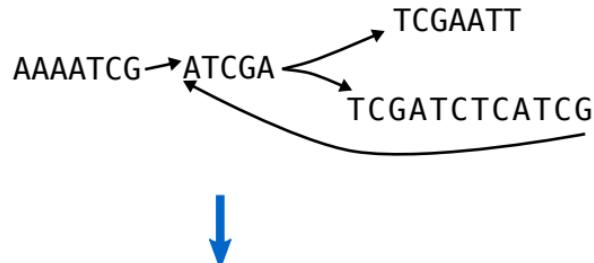
We are missing GATCTCA and ATCTCAT in the second graph.

But they are present in the first graph!

- Multiple  $k$  assembly

compacted de Bruijn graph

$k=5$



k-mers  $k=7$   
from cdBG  $k=5$

AAAATCG

AAATCGA

TCGATCT  
CGATCTC  
GATCTCA  
ATCTCAT  
TCTCATC  
CTCATCG



k-mers  $k=7$   
from the reads

AAAATCG TCTCATC  
AAATCGA CTCATCG  
AATCGAT TCATCGA  
ATCGATC CATCGAA  
TCGATCT ATCGAAT  
CGATCTC TCGAATT

compacted de Bruijn graph  
 $k=7$

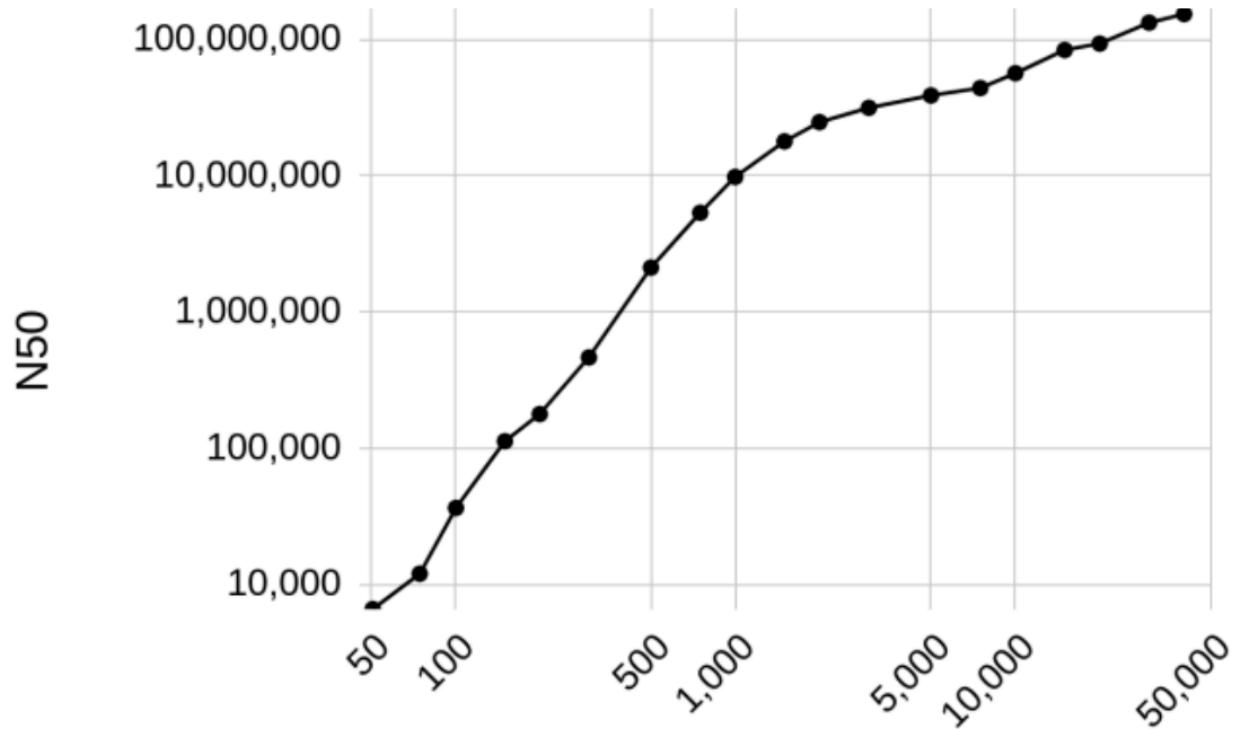


AAAATCGATCTCATCGAATT

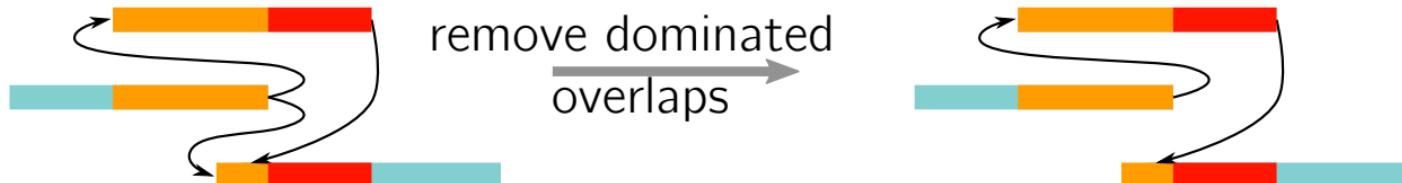
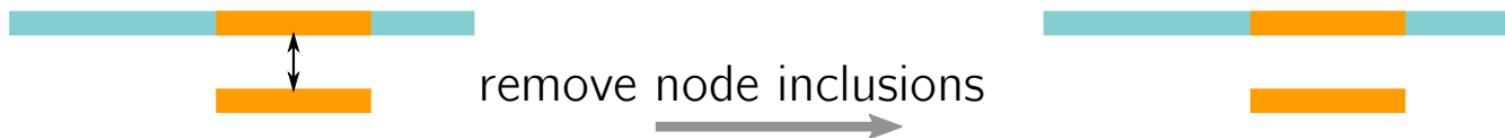
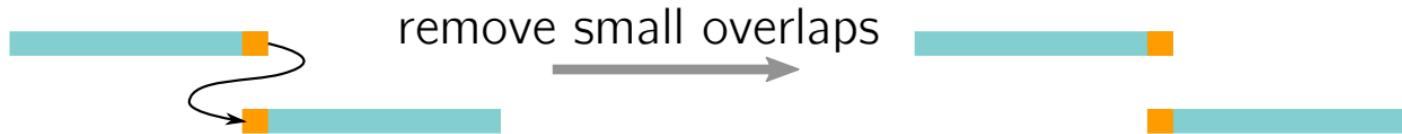
- HiFi de Bruijn graph Assembly

Using very large K (K=500 to K=5000) de Bruijn graphs to assemble

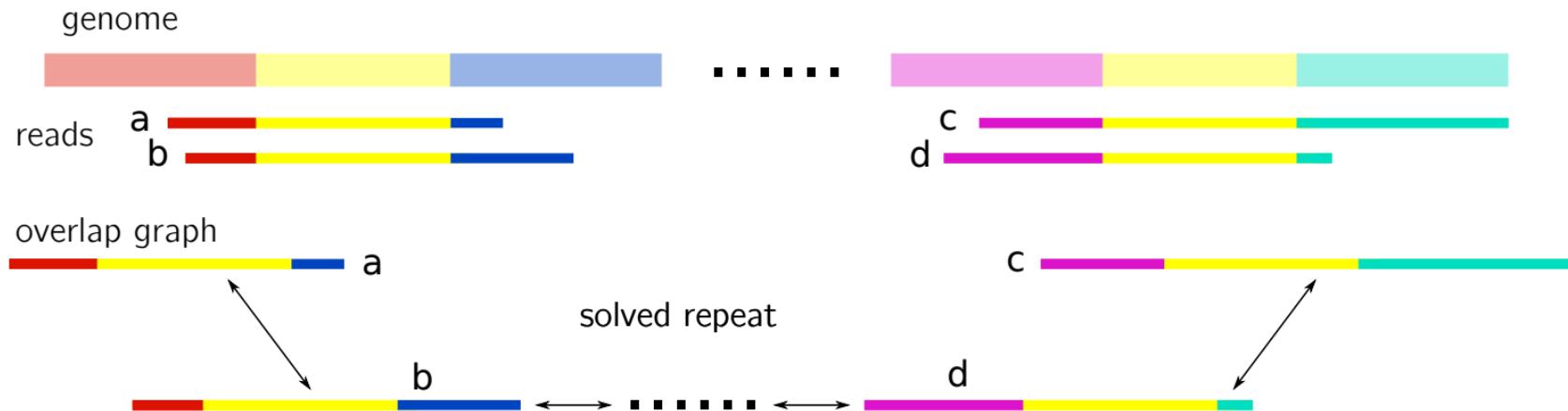
- HiFi de Bruijn graph Assembly (ii)



- Coming back to the overlap graph simplifications



- An overlap graph limitation when using noisy reads



- An overlap graph limitation when using noisy reads

genome



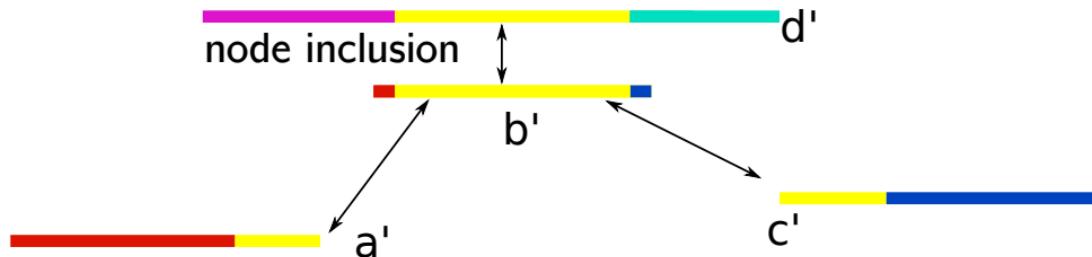
reads

a'      c'

d'     

b'

node inclusion



- An overlap graph limitation when using noisy reads

genome



a'

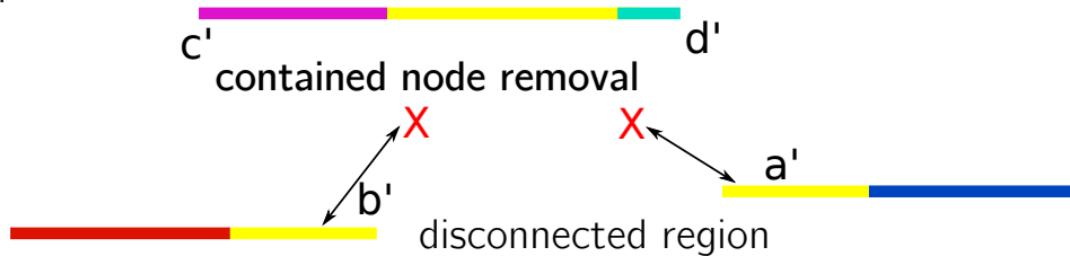
c'

d'

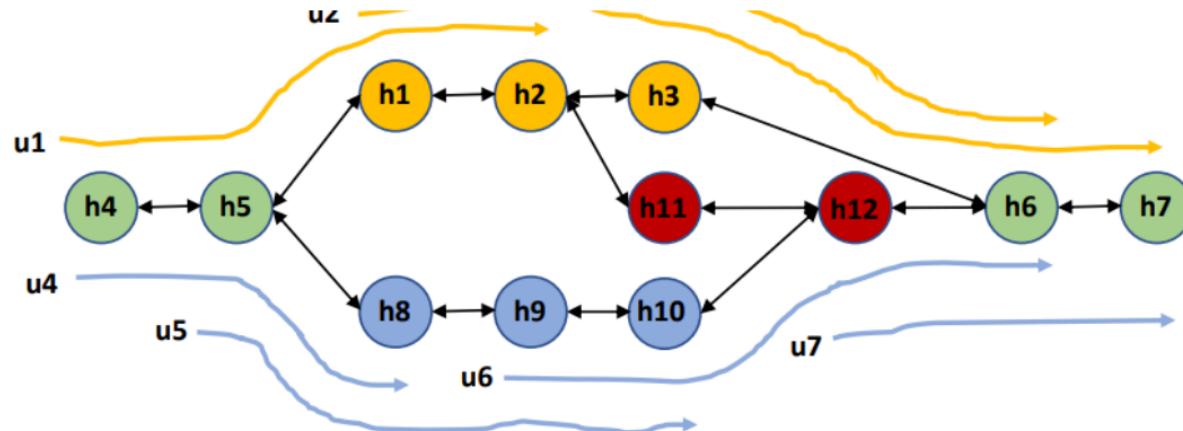
reads



overlap graph

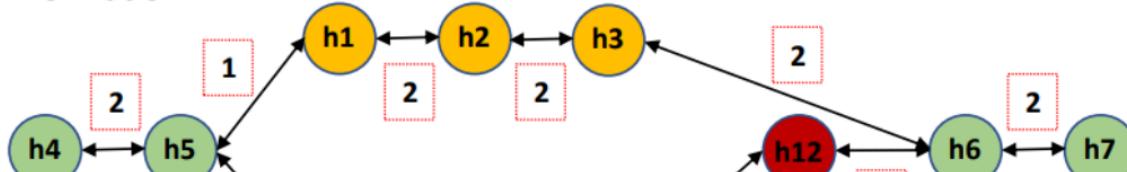


- Read threading alternative



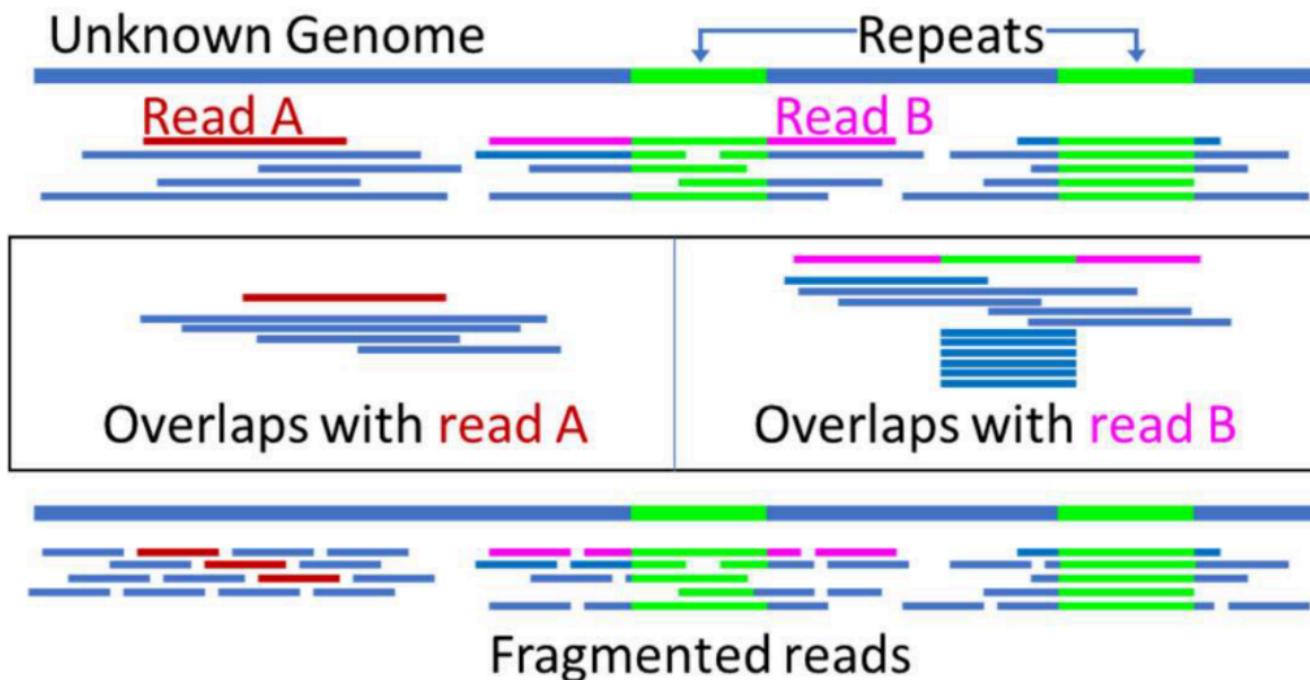
Simplify uncritical contained reads

HiFi string graph with  
ultra-long information

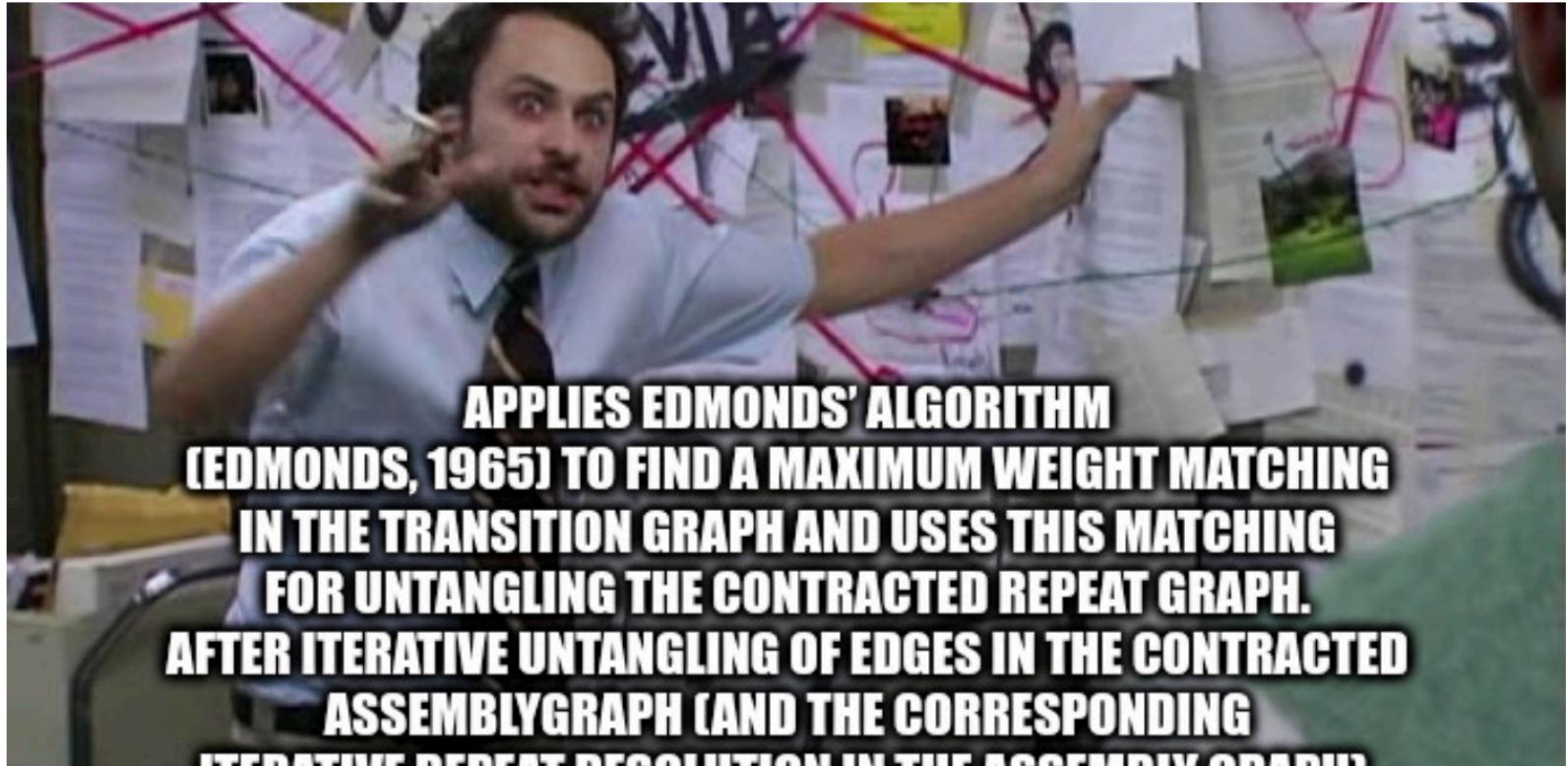


- Fragmented read alternative

The RAFT tool fragments the reads that does not cover repeats to avoid read inclusion problems.



- Flye



- Repeat graph

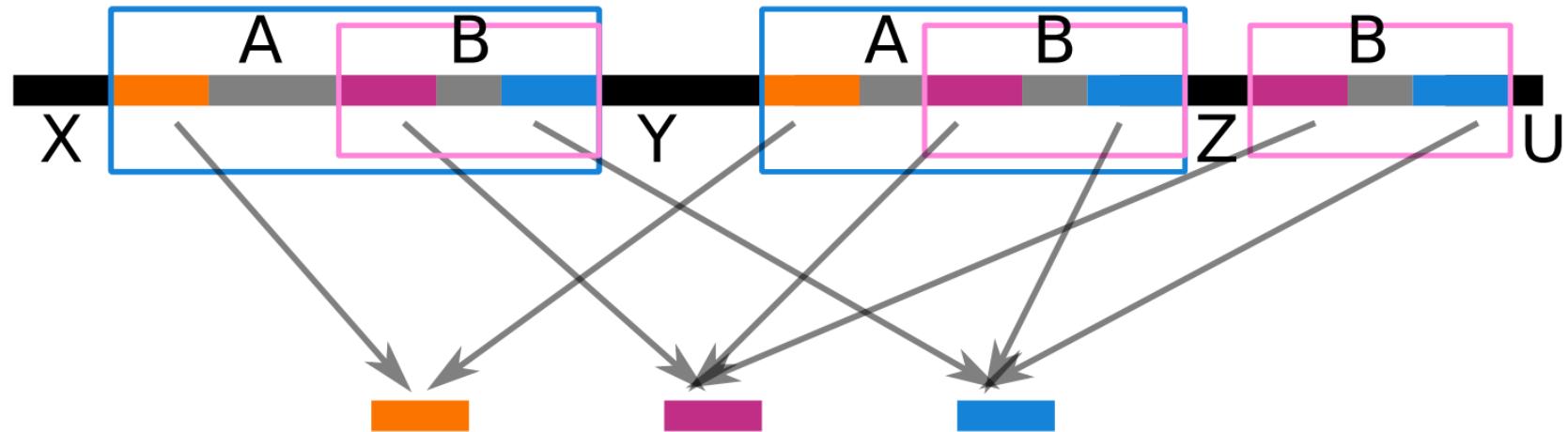
a genome



highlighted repeated regions

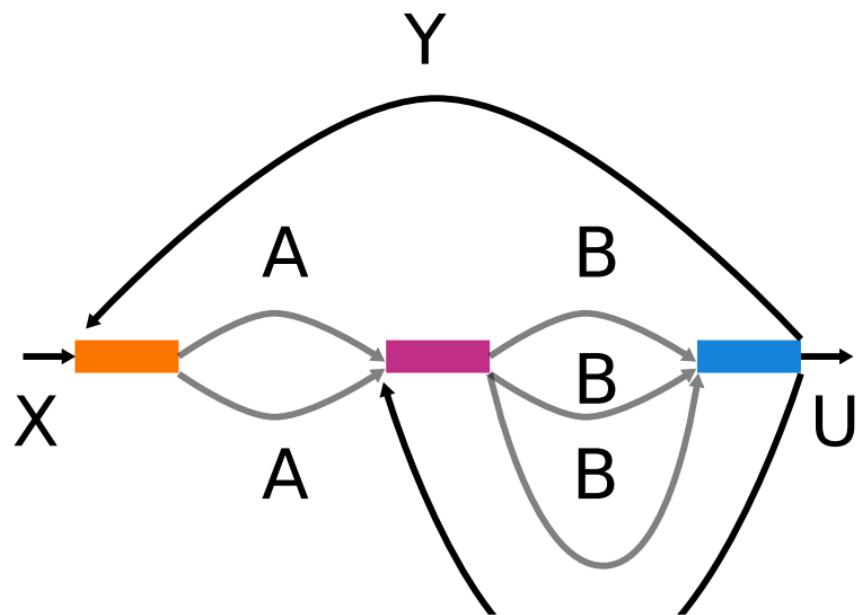


- Repeat graph

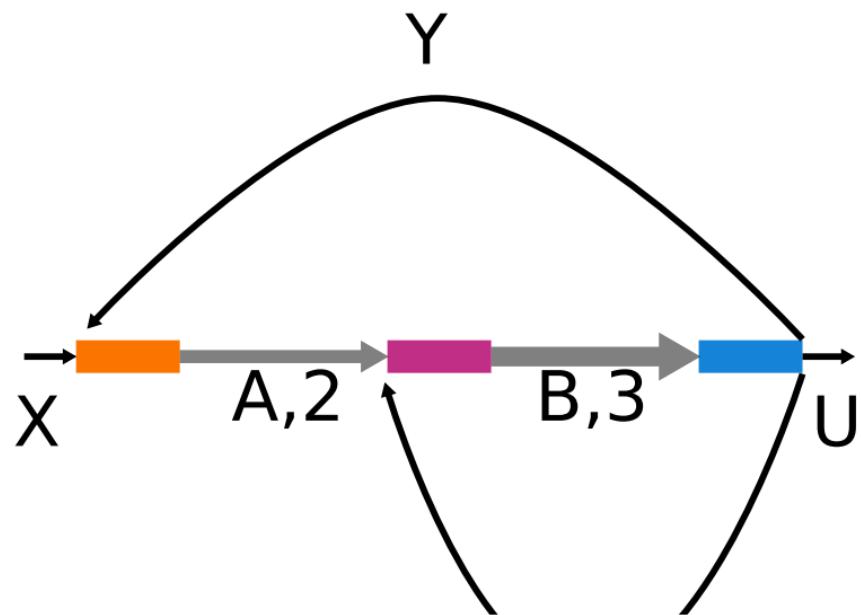


repeats extremities: graph's nodes

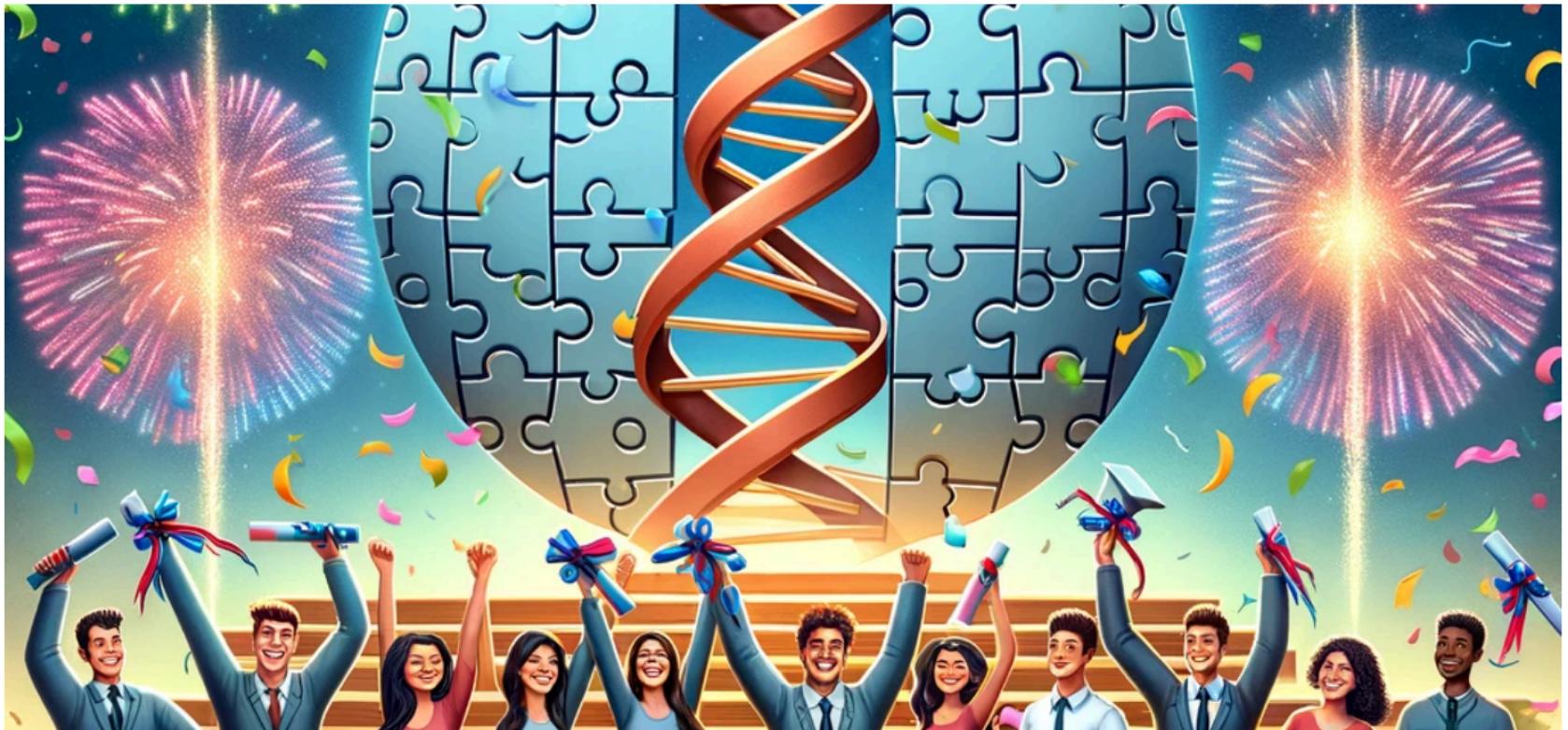
- Repeat graph



- Repeat graph



# The end (Thank you for your attention)



Slides for the practical session

- Evaluate assembly

Reference-based

body

Two cases:

**De novo**

body

## • QUAST statistics

Total aligned length	4 776 214	4 568 317	4 553 809	4 550 150
NGA50	69 801	122 647	133 309	112 446
LGA50	21	14	12	14

### Misassemblies

# misassemblies	4	0	0	4
Misassembled contigs length	231 767	0	0	435 515

### Per base quality

# mismatches per 100 kbp	2.09	2.69	1.03	3.19
# indels per 100 kbp	0.57	1.31	0.29	1.98
# N's per 100 kbp	24.59	0	17.55	94.19

### Statistics without reference

# contigs	176	95	92	90
Largest contig	248 481	235 933	285 196	264 944
Total length	4 777 853	4 571 292	4 557 363	4 552 266
Total length (>= 1000 bp)	4 757 929	4 562 458	4 548 710	4 544 453
Total length (>= 10000 bp)	4 562 801	4 478 614	4 466 223	4 475 223
Total length (>= 50000 bp)	3 248 113	3 833 793	3 812 315	3 817 904

### BUSCO completeness

- **QUAST statistics (ii)**

From <http://cab.cc.spbu.ru/quast/>

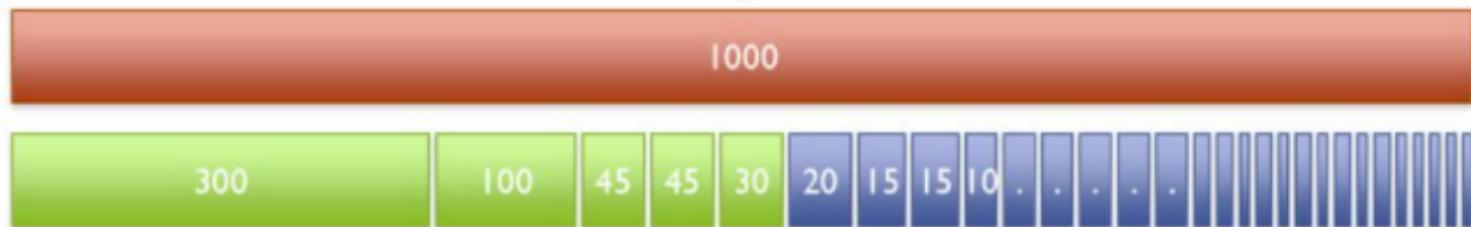
- Assembly continuity

N50

body

Example: 1 Mbp genome

50%



- The catsembler

genome

ACGGATGATAGATTGATACGA

GATTGATAC

reads    ACGGATGATA  
              TTTGATACGA

concatenate the reads: super N50!

 GATTGATACACGGATGATATTGATACGA

- **Assembly continuity**

N50

body

N75

body

NGA50

body

- Misassemblies

**Contig**



**Reference**



**Relocation**



**Inversion**



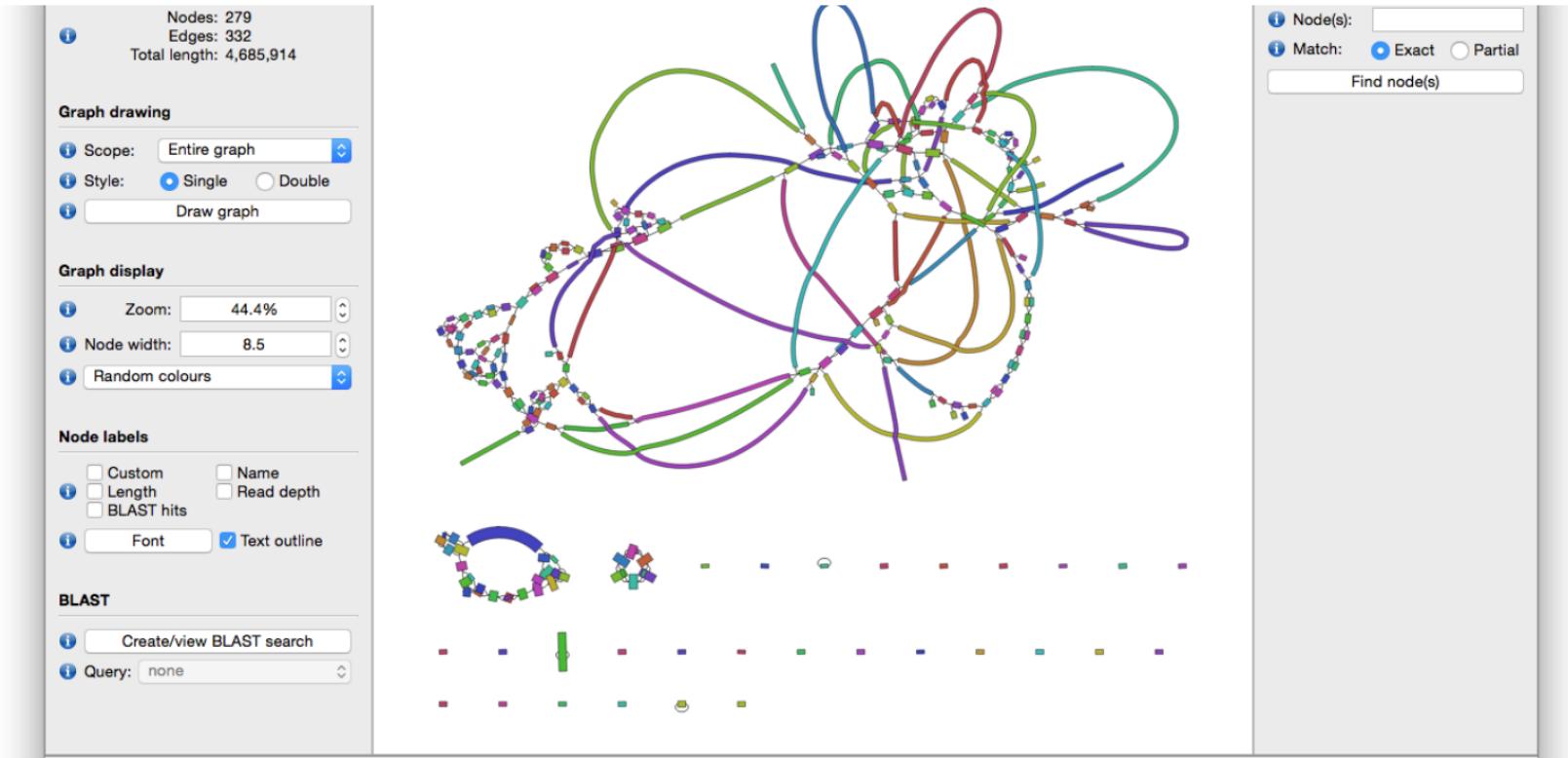
**Translocation**



- **Visualize assembly**

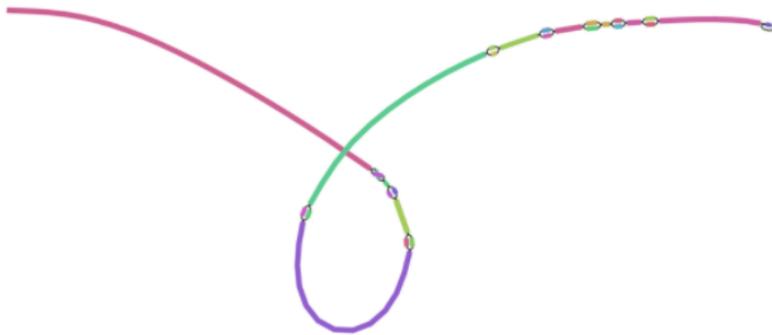
Bandage tool can visualize assembly graphs (GFA)

## • Visualize assembly (ii)

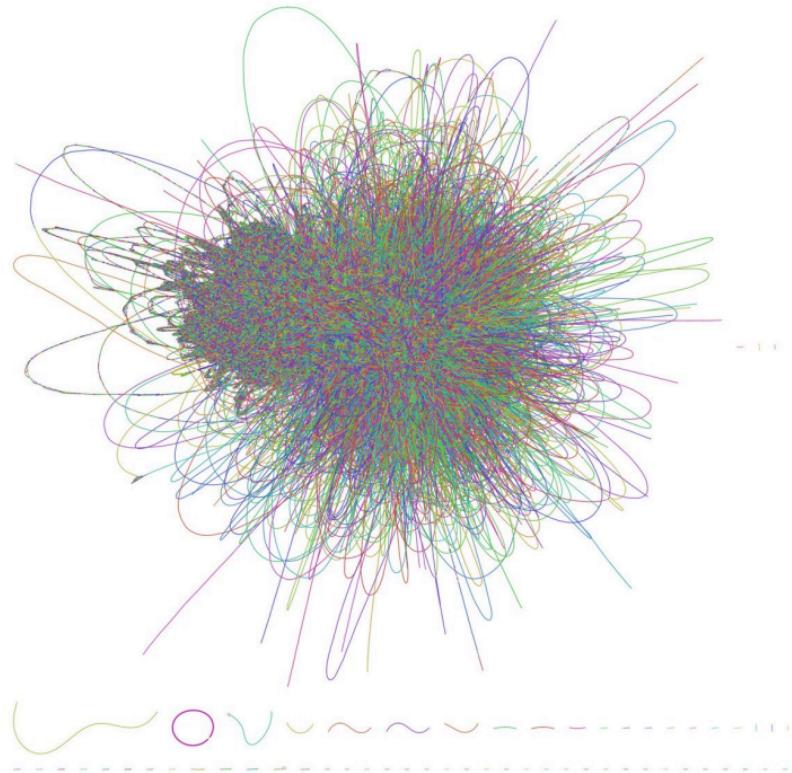


- **Visualize assembly**

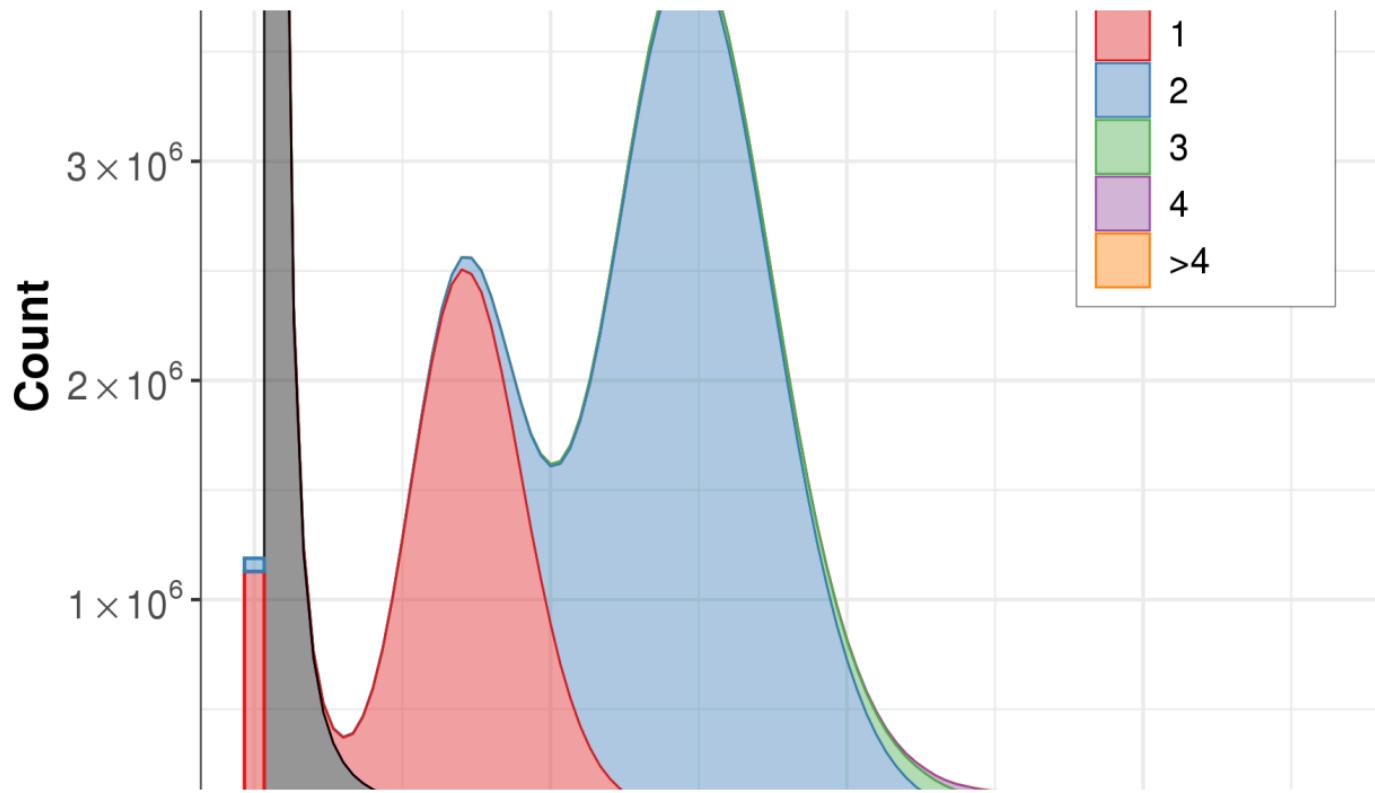
Bandage tool can visualize assembly graphs (GFA)



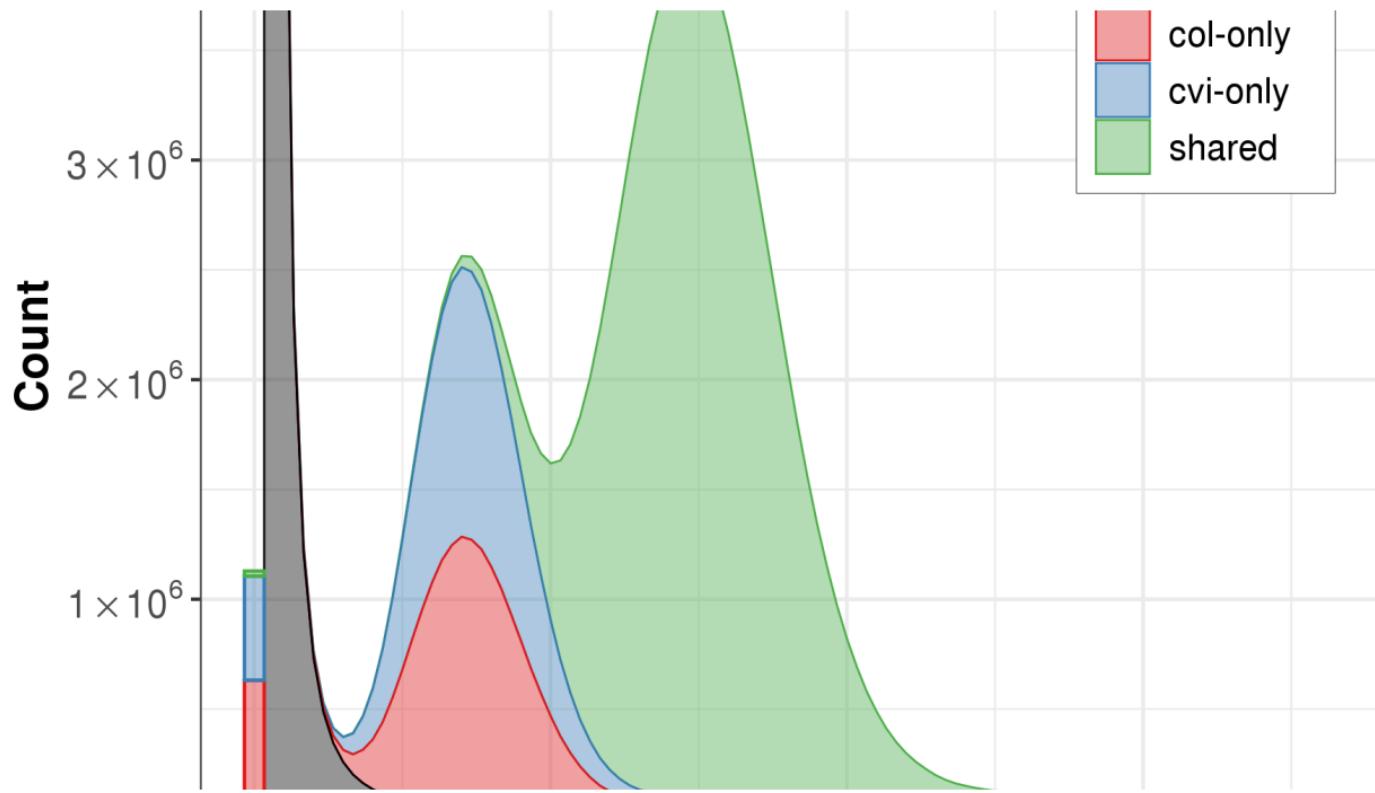
- Visualize assembly (ii)



- **K-mer spectrum visualization with merquery**



- Trio K-mer spectrum visualization with KAT



## SPAdes assembler

- Designed to assemble megabase-sized genomes
- Multiple k de Bruijn graph assembly from short reads
- Can use long reads to solve repeats

**Mandatory**

body

**Optional**

body

## Hifiasm assembler

- Build an overlap graph from HiFi reads
- Generate both haploid and diploid assemblies
- Can use (very) long reads to solve repeats

**Mandatory**

body

**Optional**

body

## Flye assembler

- Build a repeat graph from long reads
- Can use any kind of long reads
- 

**Mandatory**

body

Can also assemble metagenomes

**Optional**

body

## Unicycler (long read mode)

- Build a overlap graph from long reads
- Polish the assembly
- 

Mandatory

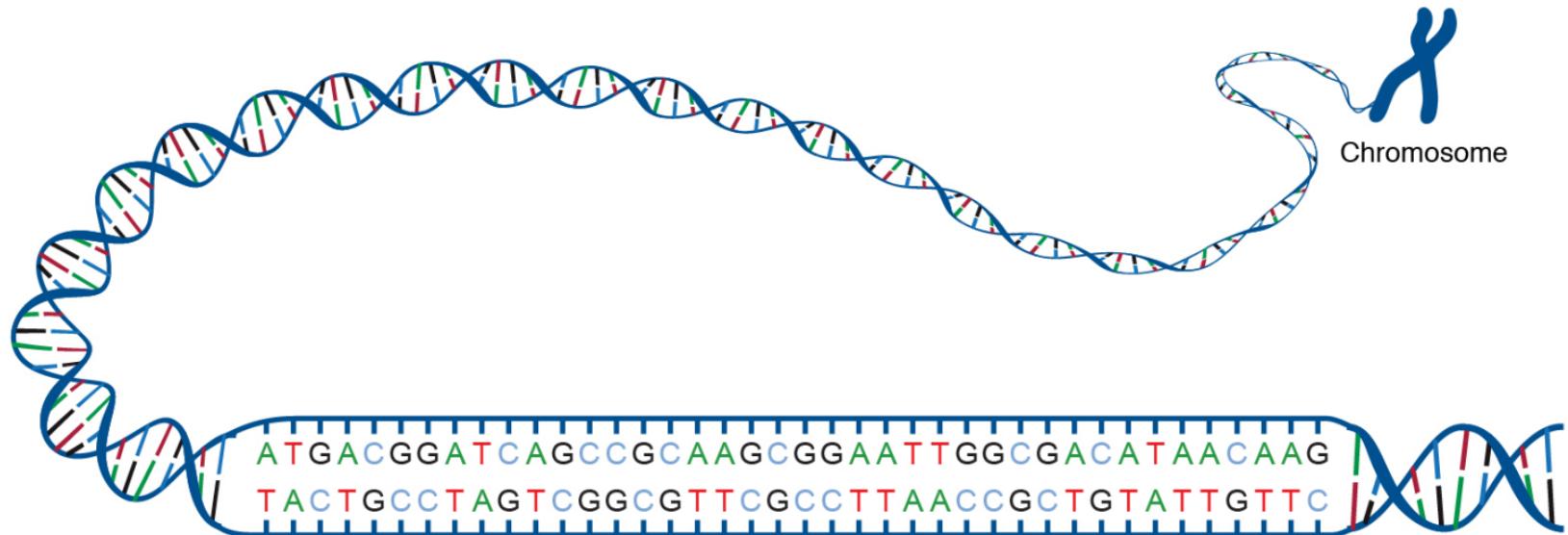
body

Also has a short-reads-first similar to SPAdes

Optional

body

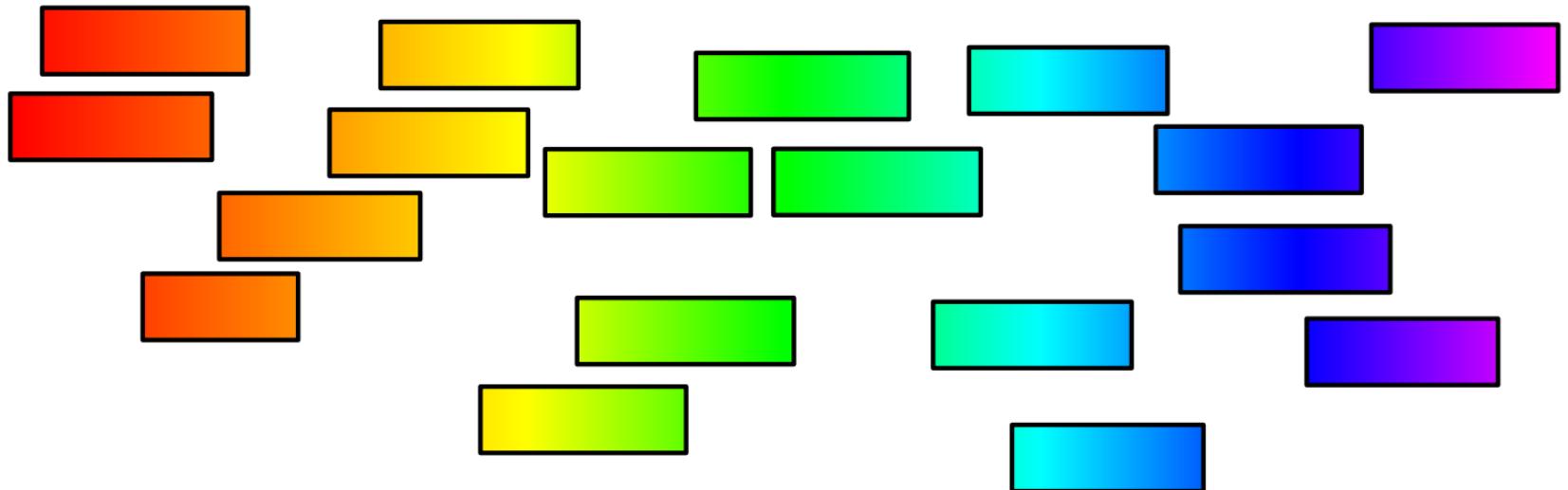
- Accessing a genome



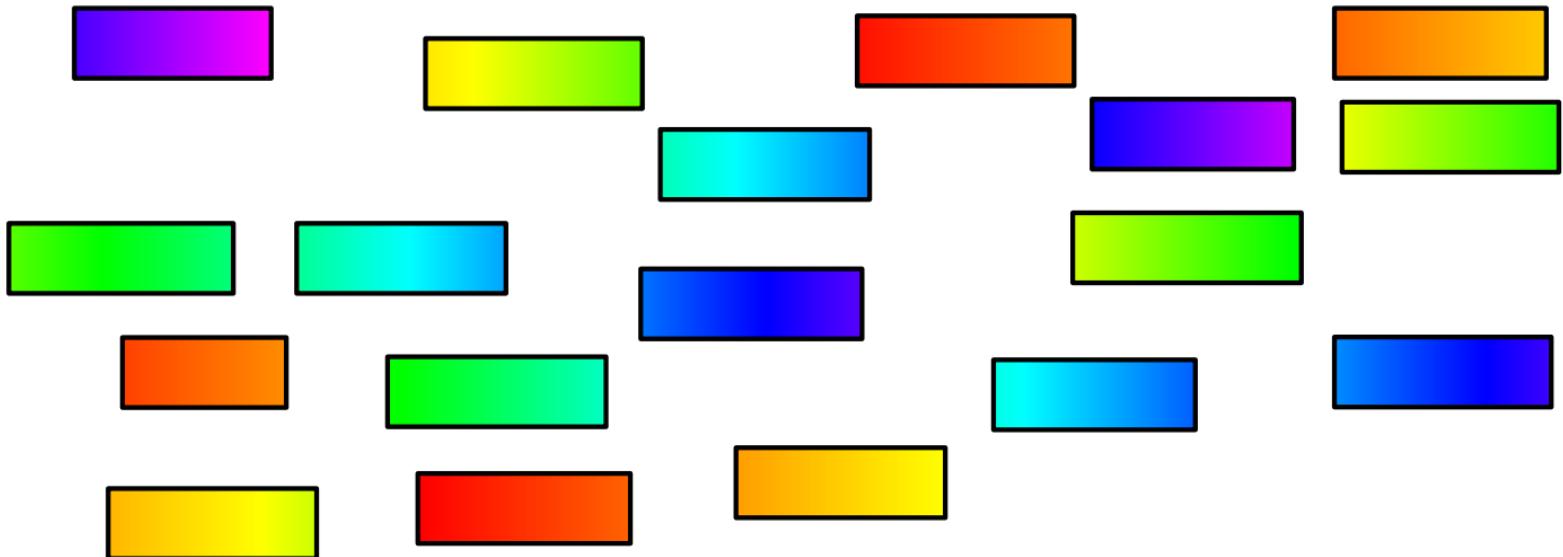
- Accessing a genome (ii)

From <https://www.genome.gov/genetics-glossary/acgt>

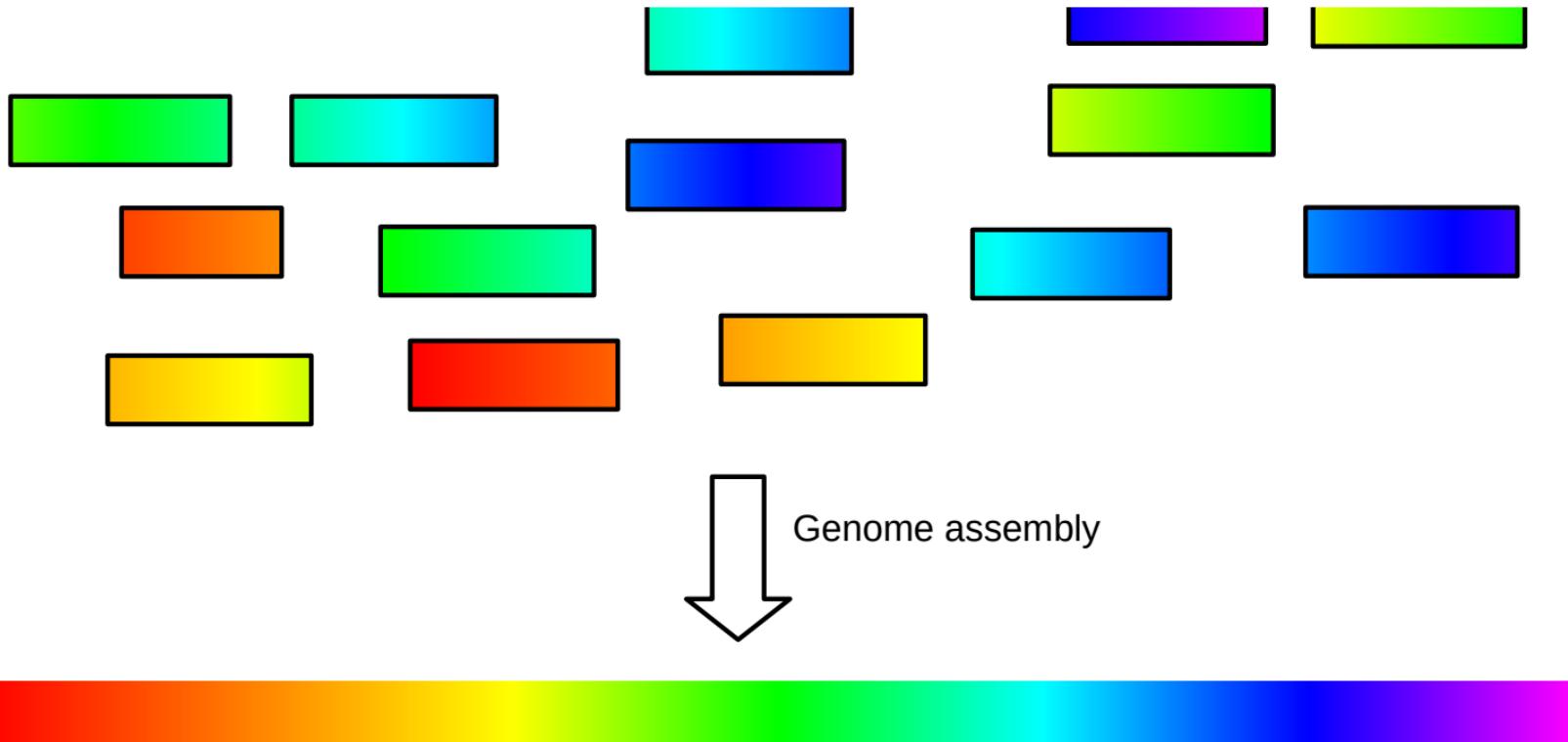
- Reads are subsequences from the genome



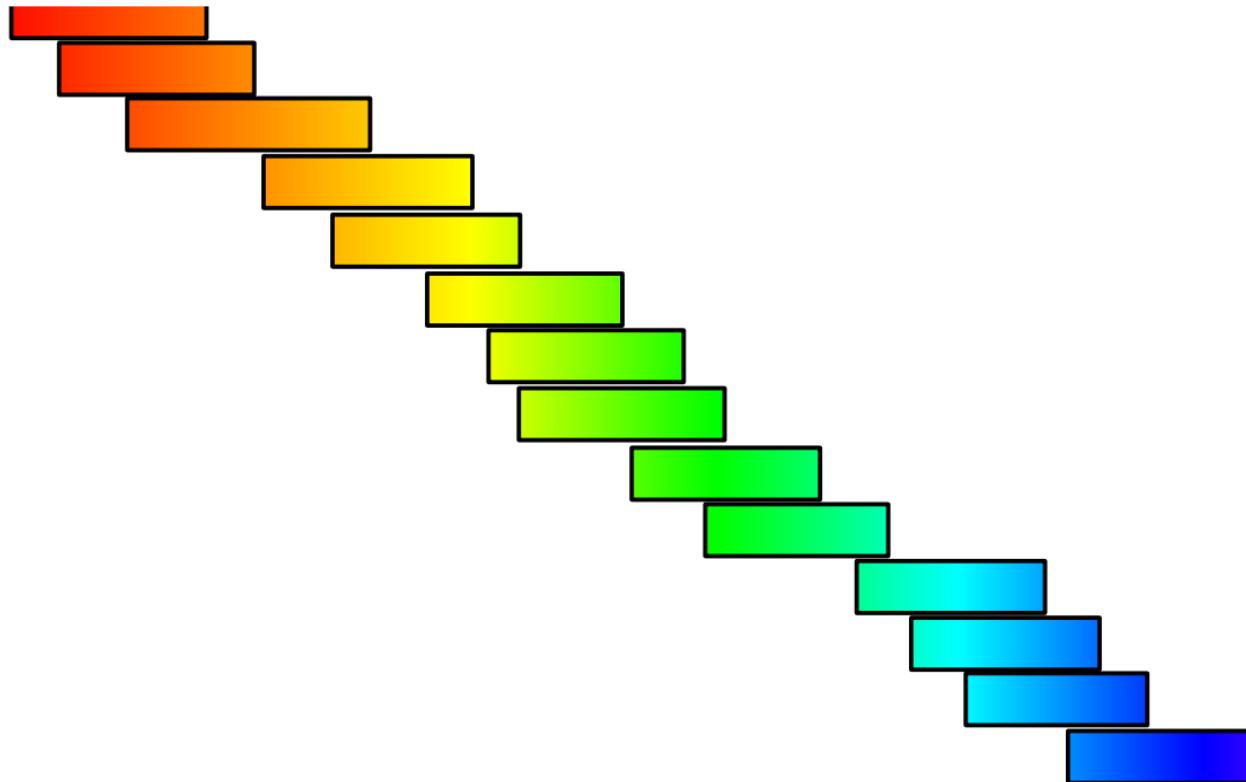
- Reads are shuffled subsequences from the genome



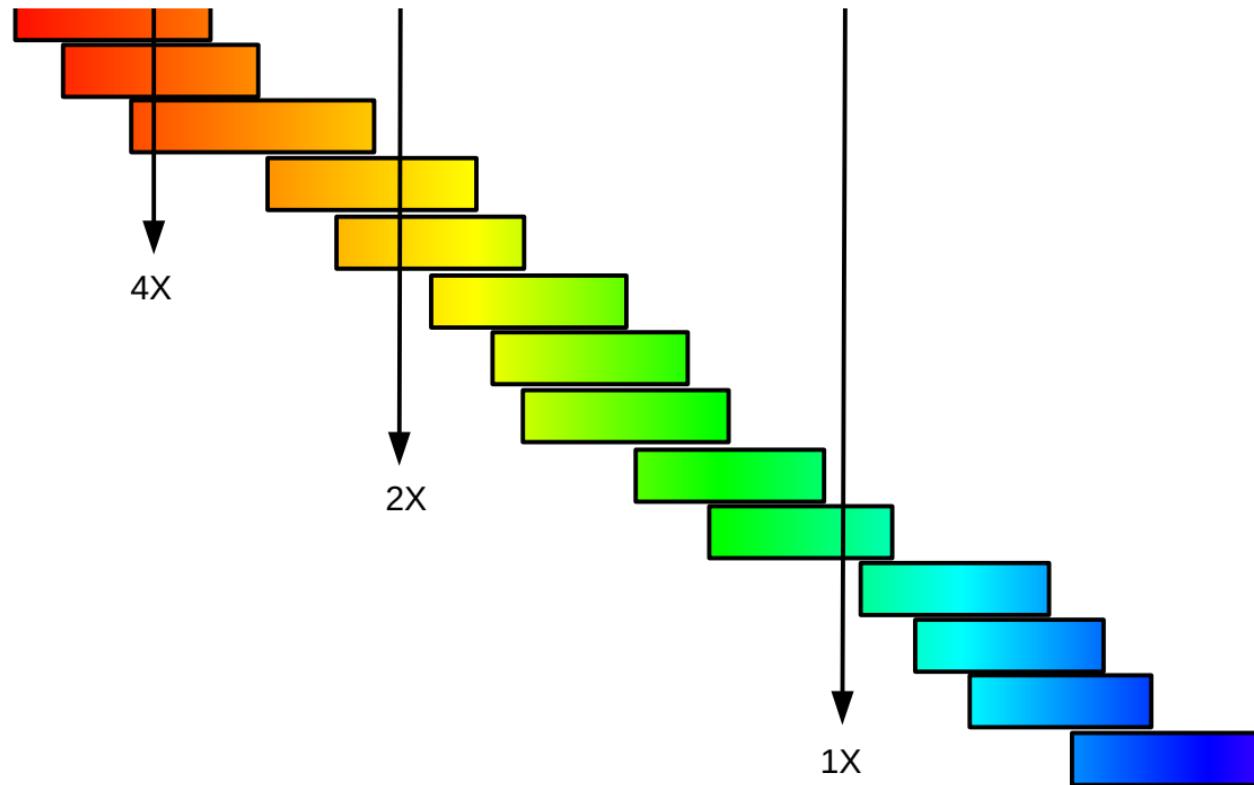
- Genome assembly task



- Using read overlaps

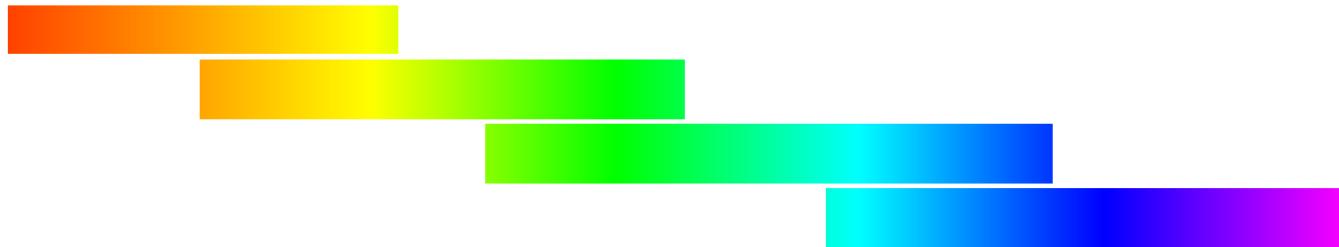


- Genome sequencing: coverage

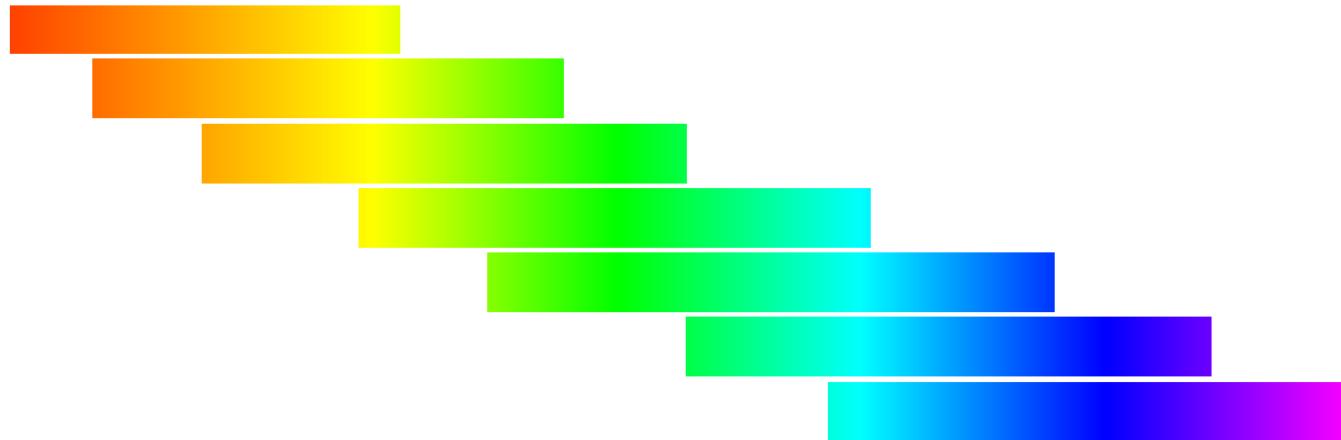


- Genome sequencing: coverage

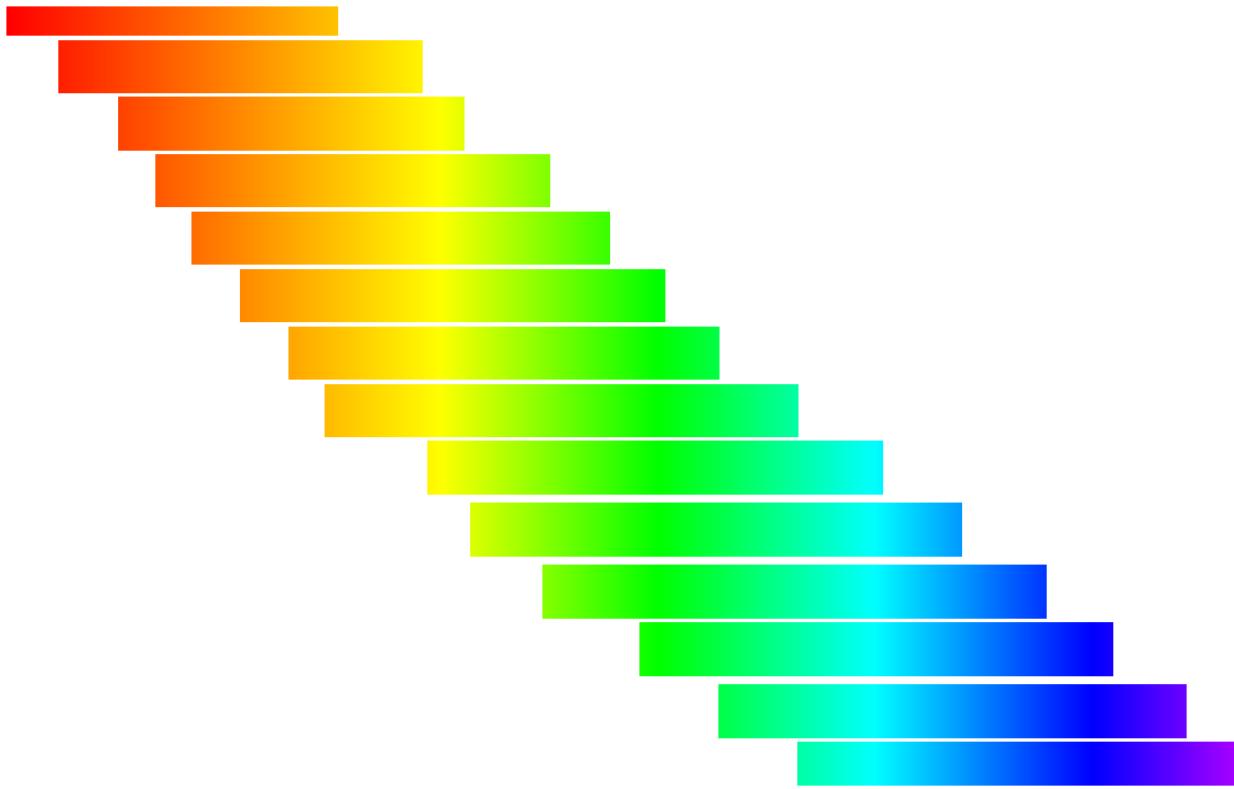
- Genome sequencing: coverage



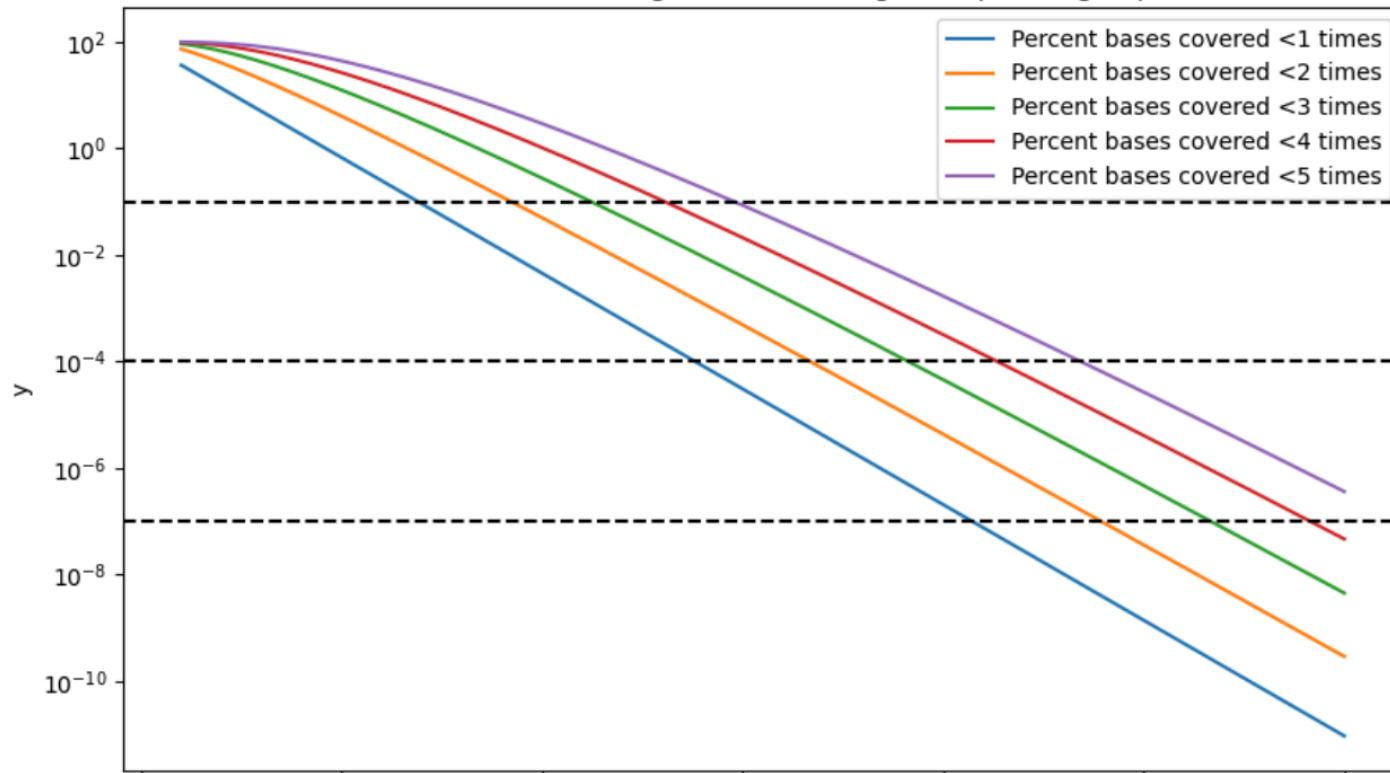
- Genome sequencing: coverage



- Genome sequencing: coverage



- Genome sequencing: coverage



- **Genome sequencing: coverage (ii)**

30-100X are often required for assembly projects

- Which overlap?

5: ACGGTATACC

1: ATC**GGTATCG**

Overlap length: 7

2: **GGTATCGTTA**

1: AT**CGGTATCG**

Overlap length: 4

3: **ATCGTTACGG**

1: AT**CGGTATCG**

Overlap length: 1

4: **GTTACGGTAT**

1: ATCGGTATACC

No Overlap

- Assembly idea number 1: assemble the longest overlaps

3: **A**T**C**GGT**A**T**C**G

4: **G**TT**A**CG**GG**T**A**T

5: **A**CG**GG**T**A**T**AC**C

Best overlaps:

1: **A**TC**GG**T**A**T**CG**

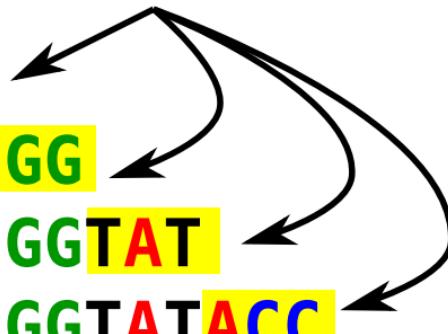
2: **GG**T**A**T**CG**TTA

3: **A**TC**G**TT**A**CG**GG**

4: **G**TT**A**CG**GG**T**A**T

5: **A**CG**GG**T**A**T**AC**C

supplementary  
information  
brought by  
each read



- Your time to shine!

Your read set:

1: ATTTC**A**CGGGT  
2: TT**A**CGGGT**GG**  
3: **A**CGGGT**C**CTT  
4: G**T**CCTTT**C**TT  
5: TTT**C**TT**A**CGG

For each read:

Find the best overlap (length>5)  
Merge the two reads

- The Greedy solution

- - - - -  
**ACGGGTCC TT**  
**GTCCTTTCTT**  
**TTTCTTACGG**

Output “genome”

**ATTTACGGGTGG**  
**ACGGGTCC TT TACGG**

- The actual solution

The actual genome:

ATTTACGGGTCCCTTTCTTACGGGTGG

How the reads should be ordered:

ATTTACGGGT  
ACGGGTCCCTT  
6 GTCCCTTTCTT  
TTTCTTACGG  
TTACGGGTGG

longest overlap we found

ATTTACGGGT  
TTACGGGTGG

8

- What happened?

The actual genome:

ATTTACGGGTCCCTTCTTACGGGTGG

How the reads should be ordered:

ATTTACGGGT  
ACGGGTCCCTT  
6 GTCCTTTCTT  
TTTCTTACGG  
TTACGGGTGG

longest overlap we found  
~~ATTTACGGGT  
TTACGGGTGG~~  
8

ATTTACGGGTGG  
not in the genome  
ACGGGTCCCTTTCTTACGG  
not in the genome

- Do we expect many repeats?

Probability to have NO repeated word of size 31 in a 5 megabases genome

body

From [https://en.wikipedia.org/wiki/Birthday\\_problem](https://en.wikipedia.org/wiki/Birthday_problem)

- The burden of assembly: genomic repeats

Amount of repeats larger than a given size in *E. coli* genome

body

Genomic repeats are NOT random events

## • Greedy assemblers

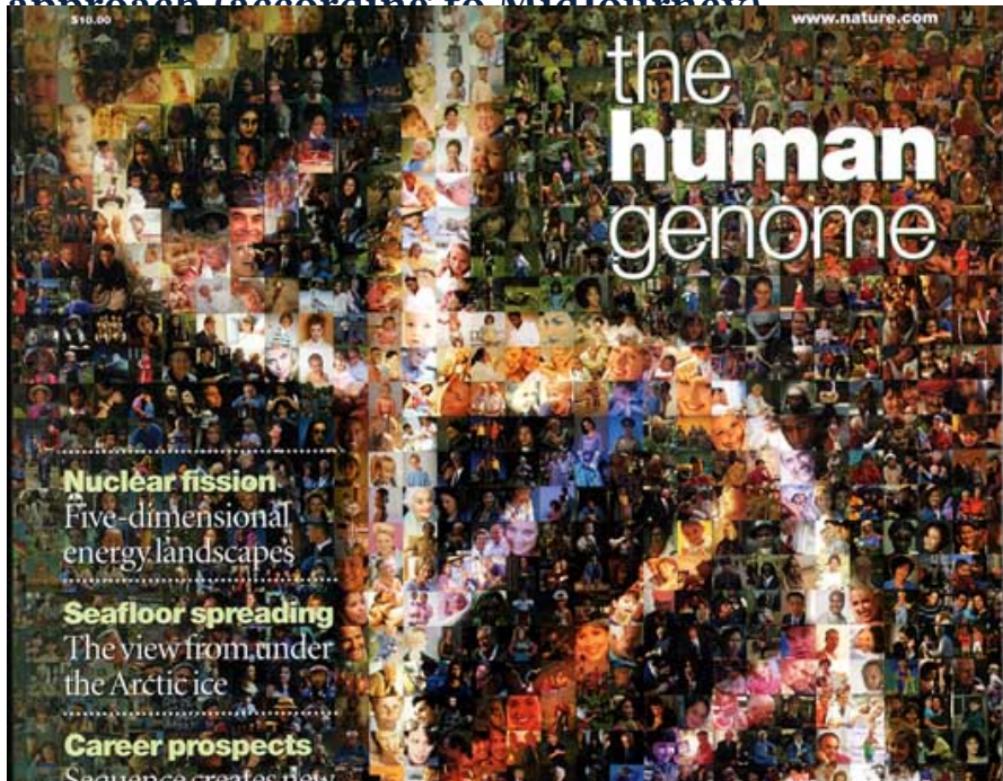
- Simple and efficient scheme
- Rely on **local** best choice (greedy)
- May create errors because of local choices when there are repeats

- Greedy assemblers (ii)



- History: the human genome project while finding a successor to the greedy approach (according to MidJourney)

245 / 419



- Graph representation

**ACGGGTCCCTT** a node is a sequence

**GTCCTTTCTT**

an arc oriented between

**TTTCTTACGG**

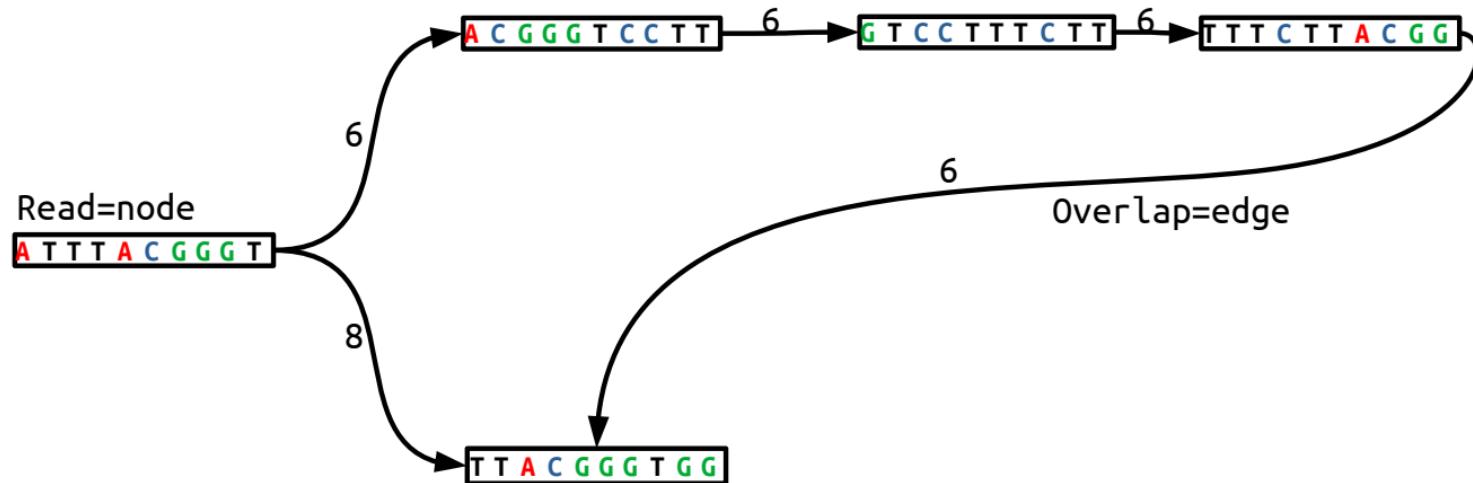
a source node and  
a sink node

an arc means there is an overlap between  
the end of the source node and the beginning  
of the sink node

- Assembly idea number 2: consider all overlaps

.....

Overlap graph:



- Greedy solution

.....

Overlap graph:



Read=node

A T T T A C G G G T

Overlap=edge

8

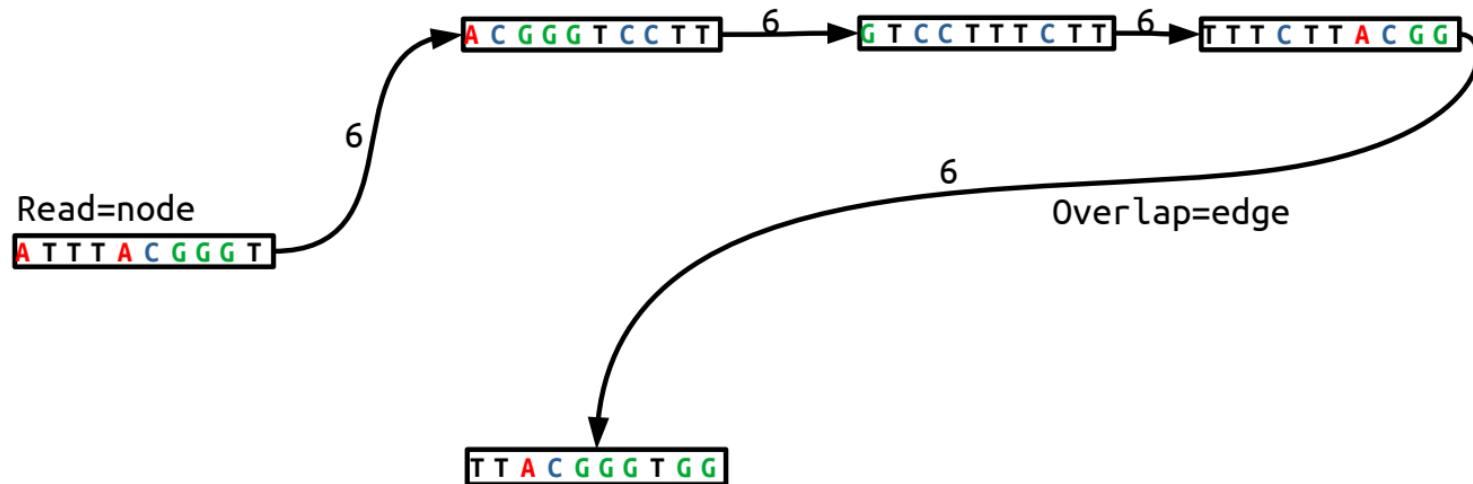
T T A C G G G T G G

Greedy assembly output:

- One piece solution

.....

Overlap graph:



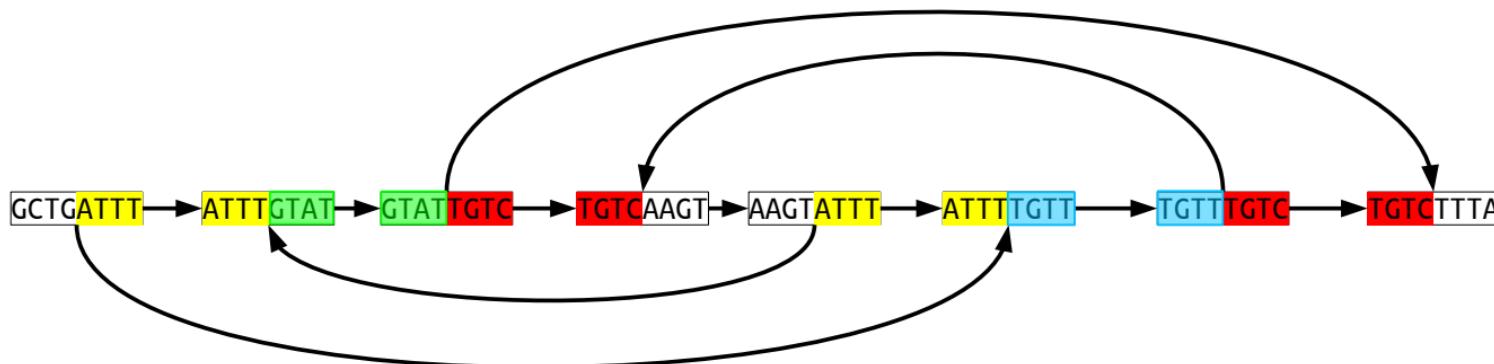
Overlap graph output:

- Multiple repeats

Reads:

GCTGATT  
ATTTGTAT  
GTATTGTC  
TGTCAAGT  
AAGTATT  
ATTTTGTT  
TGTTTGTC  
TGTCTTTA

Overlap graph:



- First solution

Reads:

GCTGATT

ATTTGTAT

GTATTGTC

TGTCAAGT

AAGTATT

ATTTTGTT

TGTTTGTC

TGTCTTA

Overlap graph:



Possible assemblies:

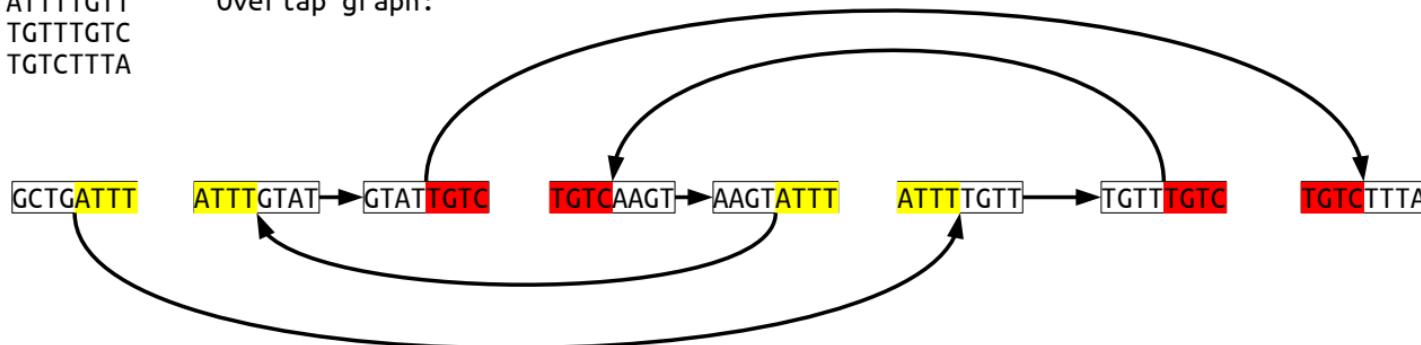
GCTGATTGTATTGTCAAGTATTTTGTTTGTCCTTA

- Second solution

Reads:

GCTGATT  
ATTTGTAT  
GTATTGTC  
TGTCAAGT  
AAGTATT  
ATTTTGTT  
TGTTTGTC  
TGTCTTTA

Overlap graph:



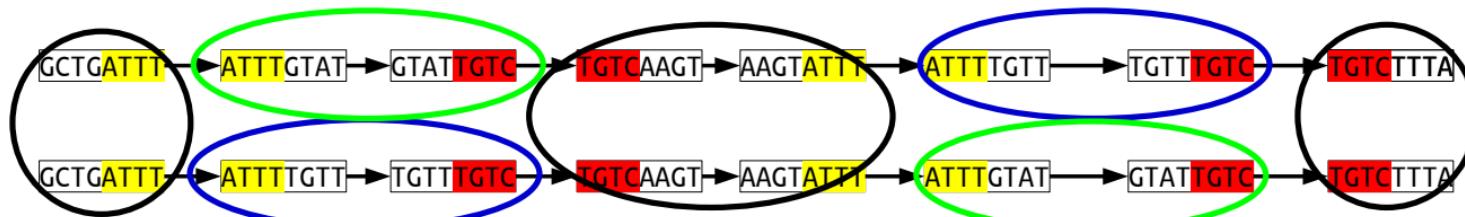
Possible assemblies:

GCTGATTGTATTGTCAAGTATTGTGGTCTTTA  
GCTGATTGTGGTCTTTA

**Those two solutions are indistinguishable**

- Parsimonious solution: do not assemble

Possible assemblies:

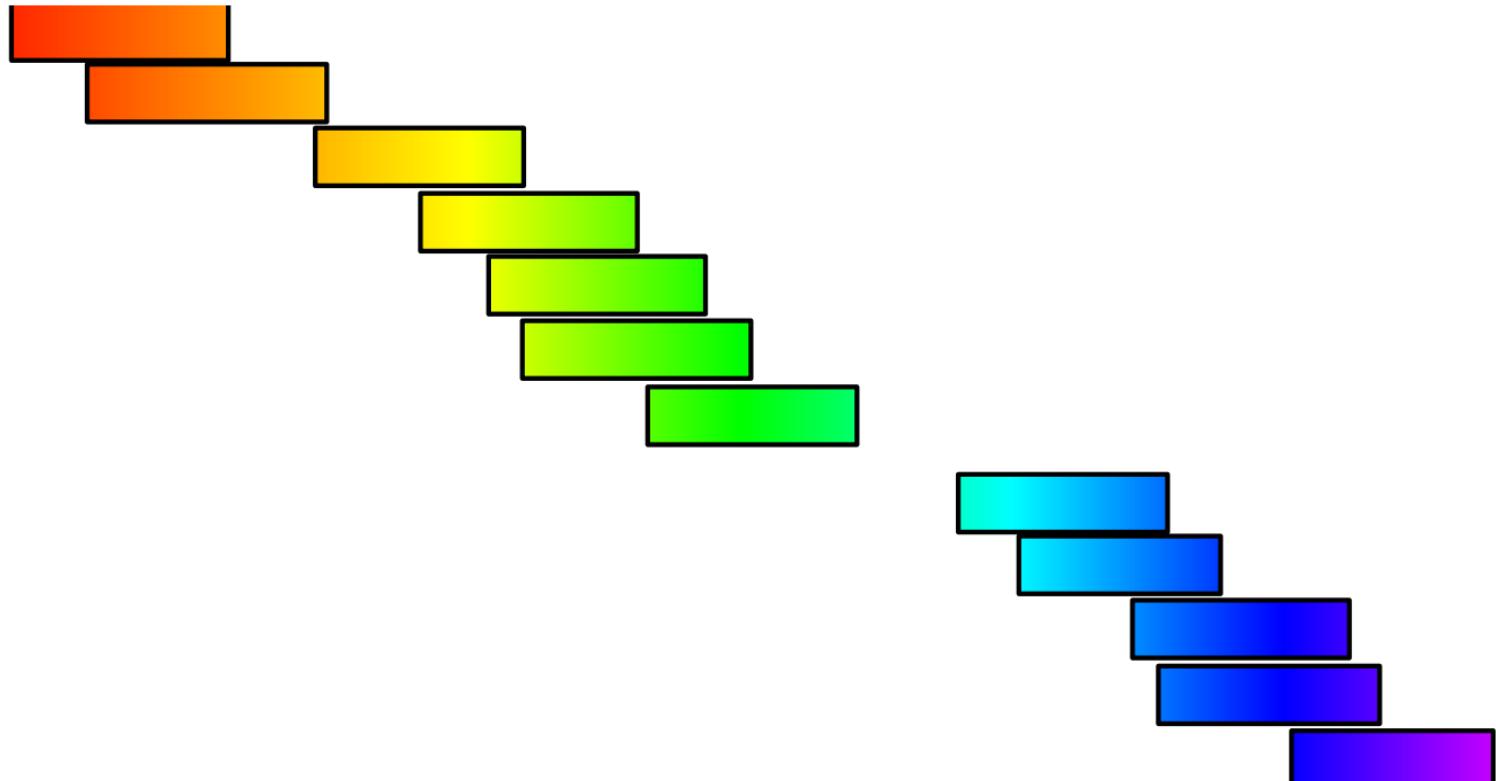


Genome pieces:

GCTGATT | ATTTGTAT TGTC | TGTC AAGTATT | ATTTGTAT TGTC |

Repeats lead to the fragmentation of the assembly

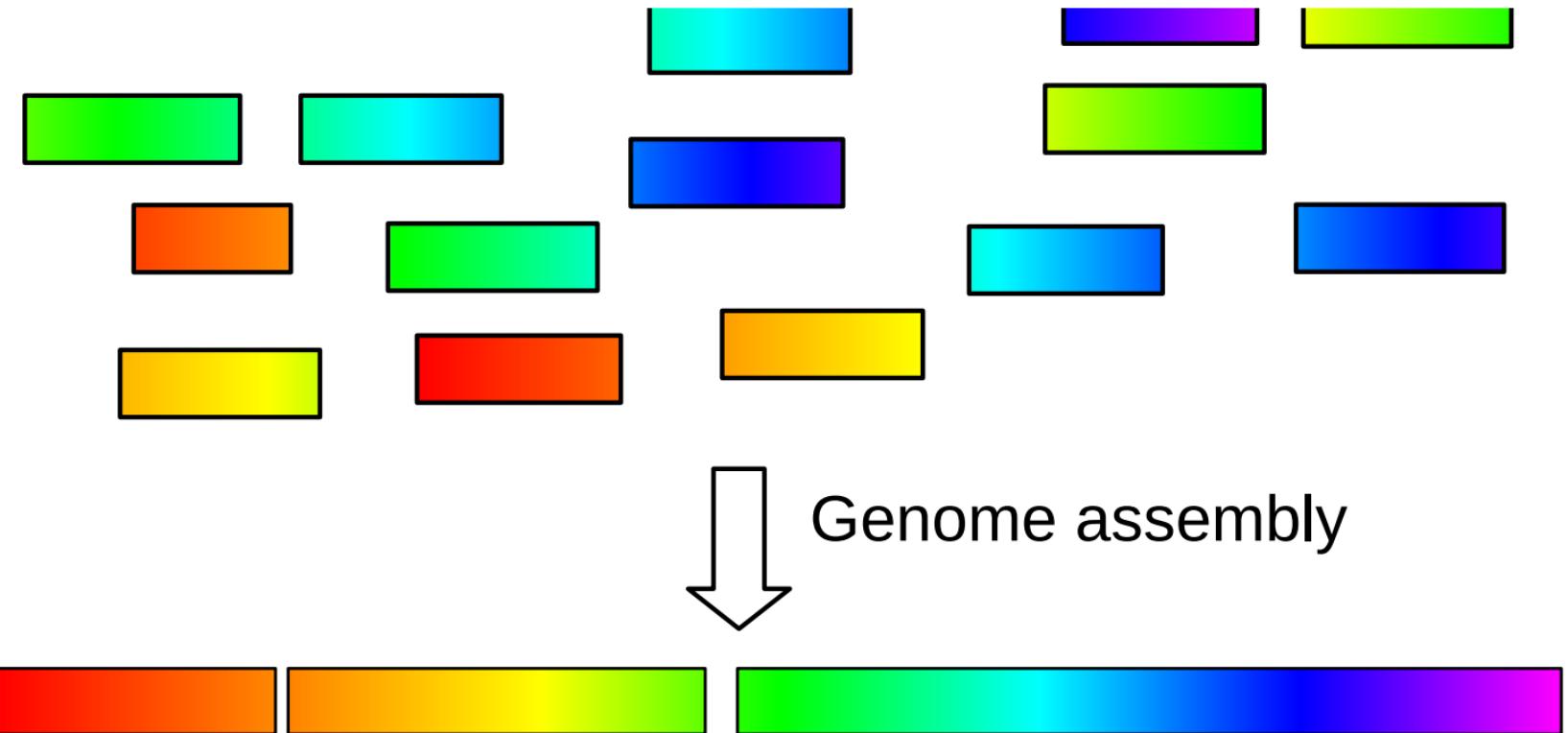
- Missing information also fragments the assembly



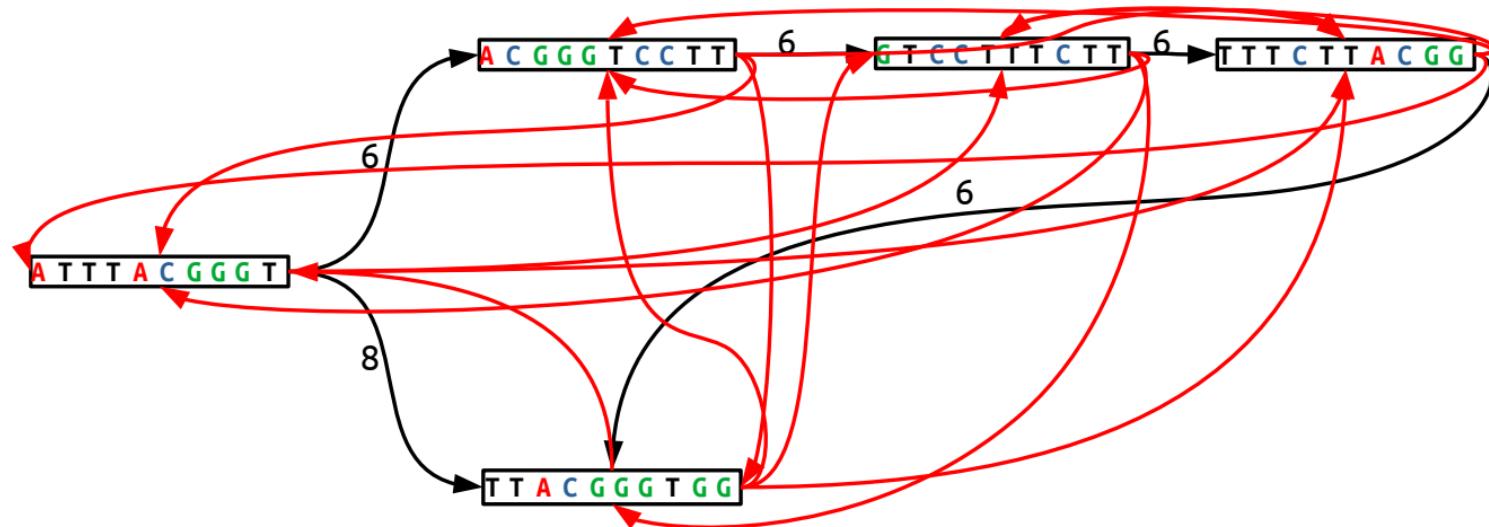
- **Assembly concession number 1: output fragments**

In the real world, assemblers often provide pieces of genomes rather than complete ones

- Assembly concession number 1: output fragments (ii)

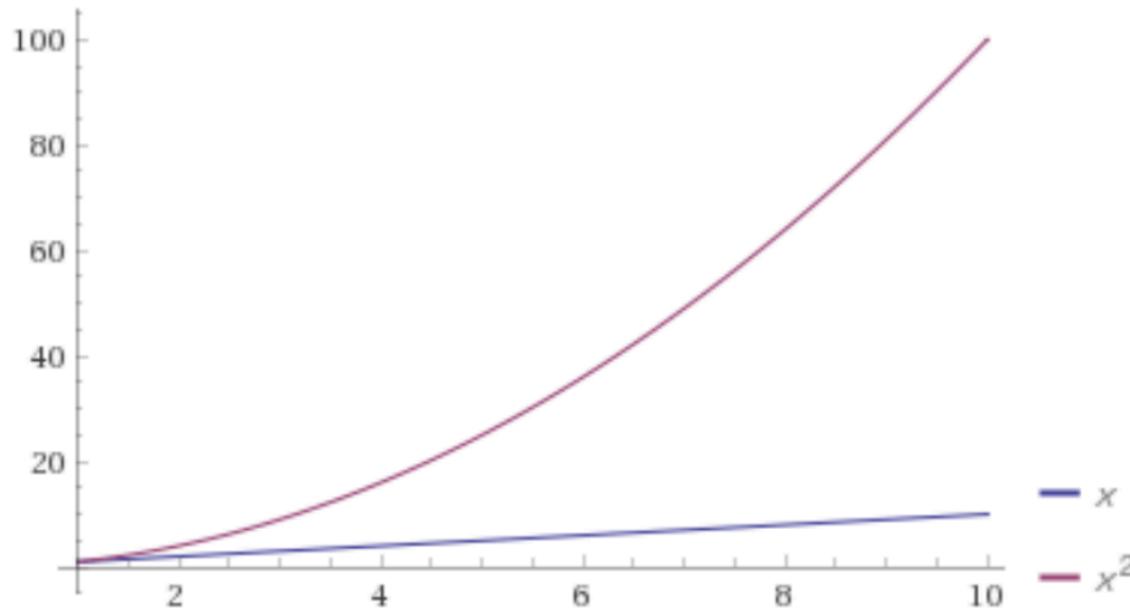


- Overlap graph prerequisite : all overlaps



- Overlap graph burden: number of reads

$$\frac{n(n-1)}{2} = O(n^2) \text{ possible overlaps for } n \text{ reads}$$



Linear: 2X data 2X time

- Overlap graph burden: number of reads (ii)

Quadratic: 2X data 4X time

- Overlap graph burden: number of reads

$\frac{n(n-1)}{2} = O(n^2)$  possible overlaps for  $n$  reads

# Reads	# Overlaps
1000	499,500
10,000	50 million
100,000	5 billion
1 million	500 billion
10 million	50 trillion...

The overlap computation is not linear

Talking about CPU years on large genomes...

We have to be efficient here and focus on “relevant” overlaps

## • Overlap graph simplifications



remove small overlaps



remove node inclusions



remove dominated  
overlaps



## • Overlap graphs in a nutshell

- Graphs of overlaps between the reads
- Can provide a global solution for assembly
- Can be difficult in real cases because it requires a lot of computation (overlaps)



*S. cerevisiae*, *D. melanogaster*, human could be assembled using overlap graphs approaches (Celera (Myers et al. 2000), SGA (Simpson & Durbin 2011), ...)

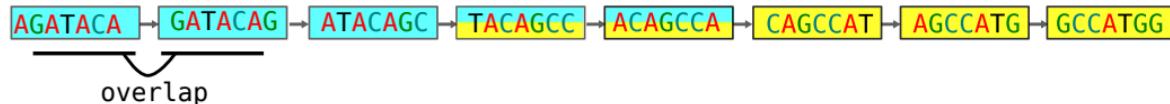
- History: the introduction of another graph structure for assembly, according to MidJourney

263 / 419

Overlapping reads

AGATAACAGCCA  
TACAGCCATGG

De Bruijn graph



Resulting sequence

AGATAACAGCCATGG

- Assembly idea number 3: Focus on genome words

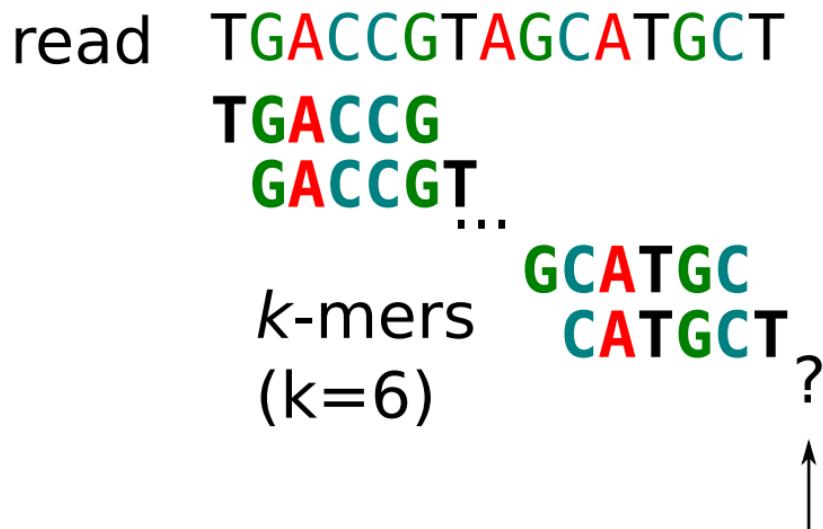
a genome: a succession of words

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC  
GCATGC  
CATGCT

- Find consecutive genome words in read words

AGATAACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

read    TGACCGTAGCATGCT  
          TGACCG  
          GACCGT...  
                  GCATGC  
                  CATGCT  
                 ?  
k-mers (k=6)



- How to connect read words?

Genomic

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

reads TGACCGTAGCATGCT 1  
GACCGTAGCATGCTA 2

$k$ -mers  
( $k=6$ )

GCATGC 1 2  
CATGCT 1 2  
ATGCTA 2

next nucleotide

- Reconstitute larger genomic words

reads

TGACCGTAGCATGCT ①  
GACCGTAGCATGCTA ②



extract the read's  
k-mers ( $k=6$ )

- The de Bruijn graph



## De Bruijn graph

Kmer=node



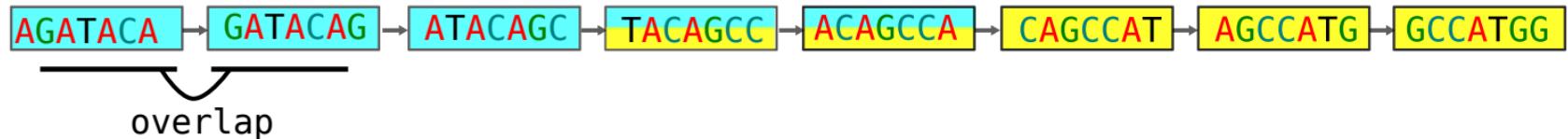
AGATA + G + C + C + A

- de Bruijn graph assembly

Overlapping reads

AGATA**CAGCCA**  
**TACAGCCATGG**

De Bruijn graph



Resulting sequence

AGATA**CAGGCCATGG**

- de Bruijn graph time!

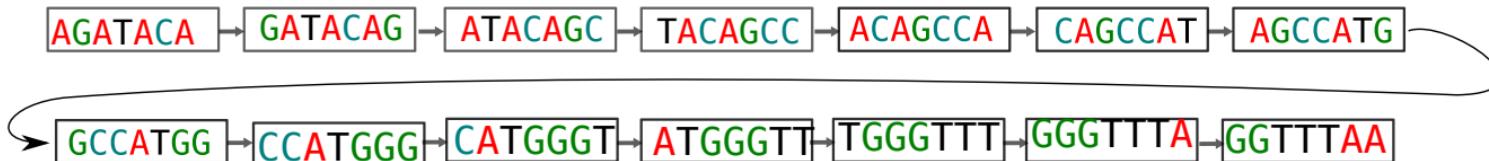
.....  
GCCATGGGTTT  
TACAGCCATGG  
AGCCATGGGTT  
GCCATGGGTTT  
AGATAACAGCCA  
ACAGCCATGGG  
GATACAGCCATG  
CATGGGTTTAA  
ACAGCCATGGG  
GATACAGCCATG  
CATGGGTTTAA  
CACCGCTTCT

Hint: Use 7-mers

- Solution

GA **TACAGCCATGG**  
 TACAGCC**ATGG**  
 ACAGCC**ATGGG**  
 ACAGCC**ATGGG**  
 CAGCC**ATGGGT**  
 AGCC**ATGGGTT**  
 GCC**ATGGGTTT**  
 GCC**ATGGGTTT**  
 CAT**GGGTTTAA**  
 CAT**GGGTTTAA**

De Bruijn graph



- de Bruijn graphs abstract redundancy

GATACAGCCATG  
 GATACAGCCATG  
 TACAGCCATGG  
 ACAGCCATGGG  
 ACAGCCATGGG  
 CAGCCATGGGT  
 AGCCATGGGTT  
 GCCATGGGTTT  
 GCCATGGGTTT  
 CATGGGTTTAA  
 CATGGGTTTAA

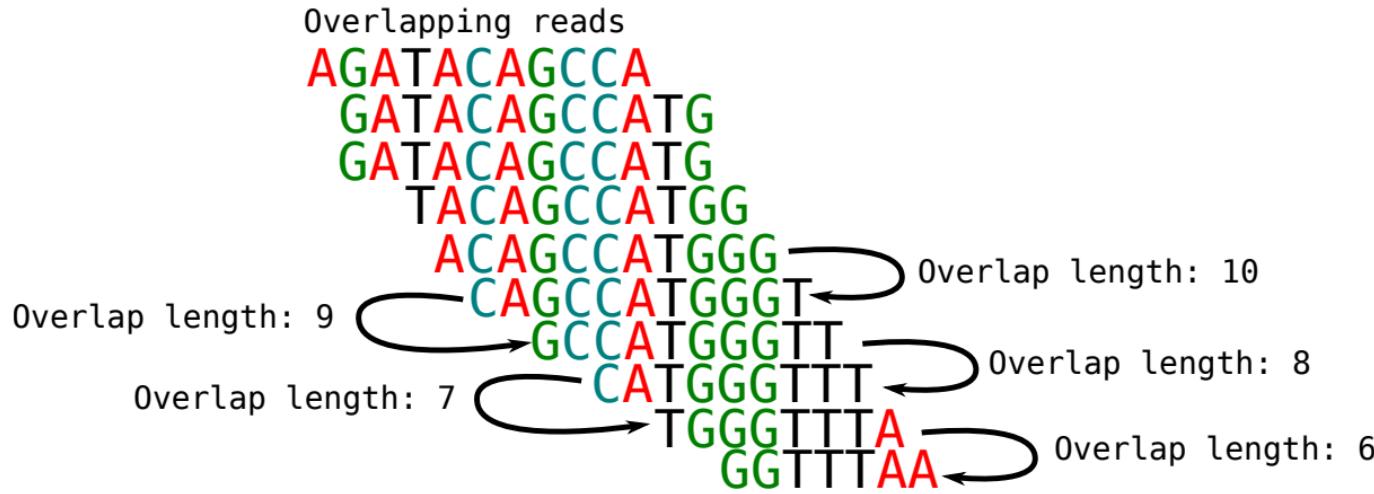
62 (**non distinct**) 7-mers in the reads

De Bruijn graph

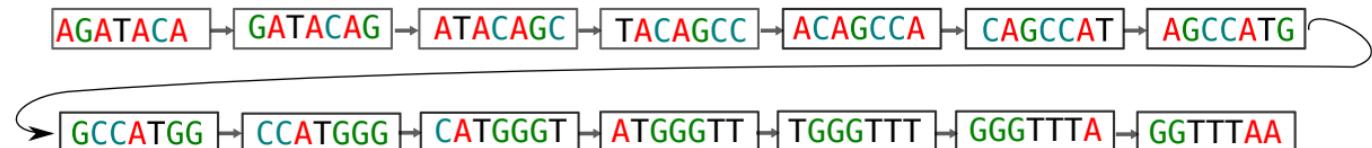
14 **distinct** 7-mers in the De Bruijn graph



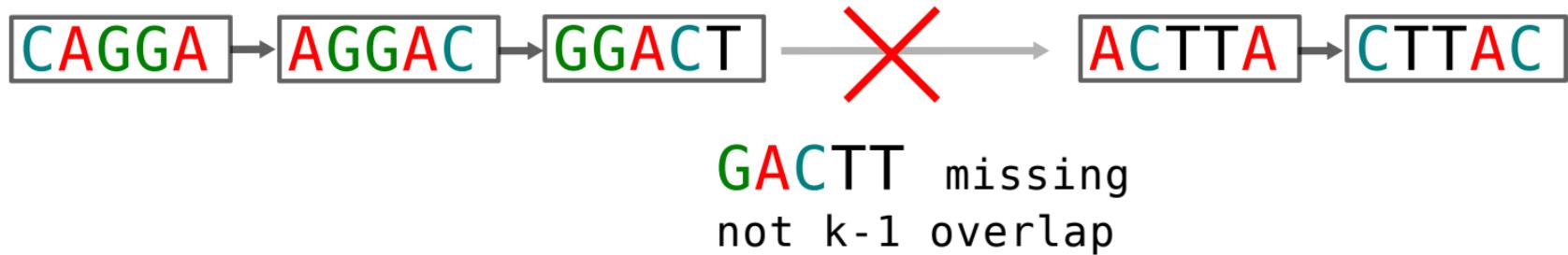
- de Bruijn graphs only rely on  $k - 1$  overlaps



De Bruijn graph overlap length: 6



- de Bruijn graphs limitation 1: Fixed overlaps



GGACT and ACTTA overlap is only of size 3 !

- de Bruijn graphs limitation 2: Repeats

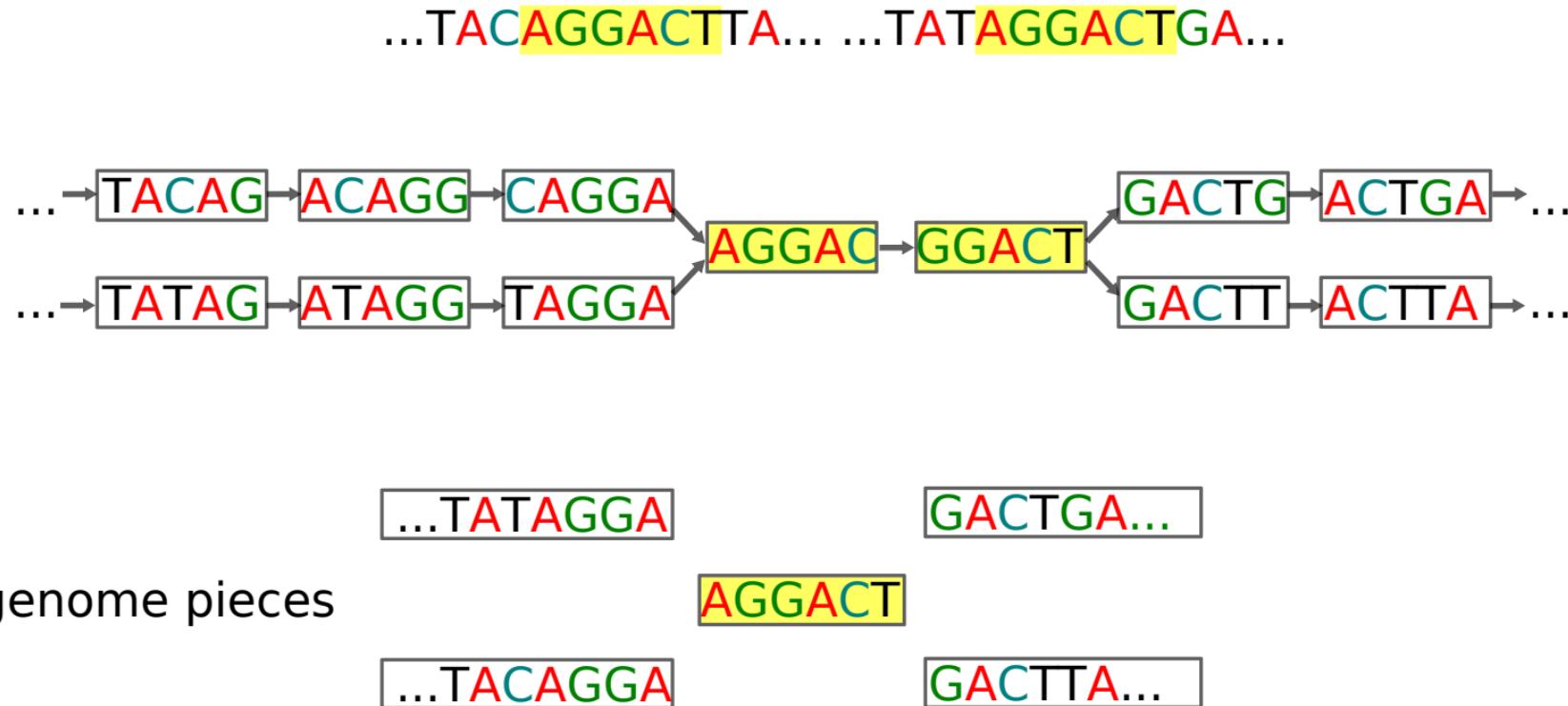
...TACAGGACTTA... ...TATAGGACTGA...



#v(0.4cm)

each  $k$ -mer appears only once in a de Bruijn graph

- de Bruijn graph limitation



- de Bruijn graph limitation (ii)

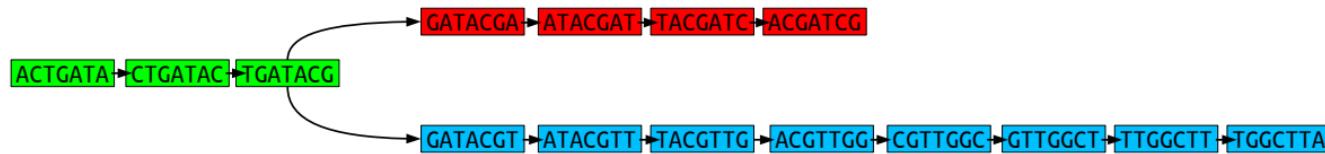
- de Bruijn graph versus overlap graph

**de Bruijn graph**:  
AGATACA → GATAACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG

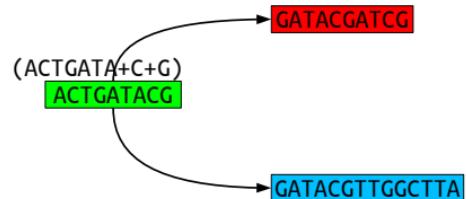
Overlap graph from the reads



- On the representation of de Bruijn graphs



Compacted De Bruijn graph:



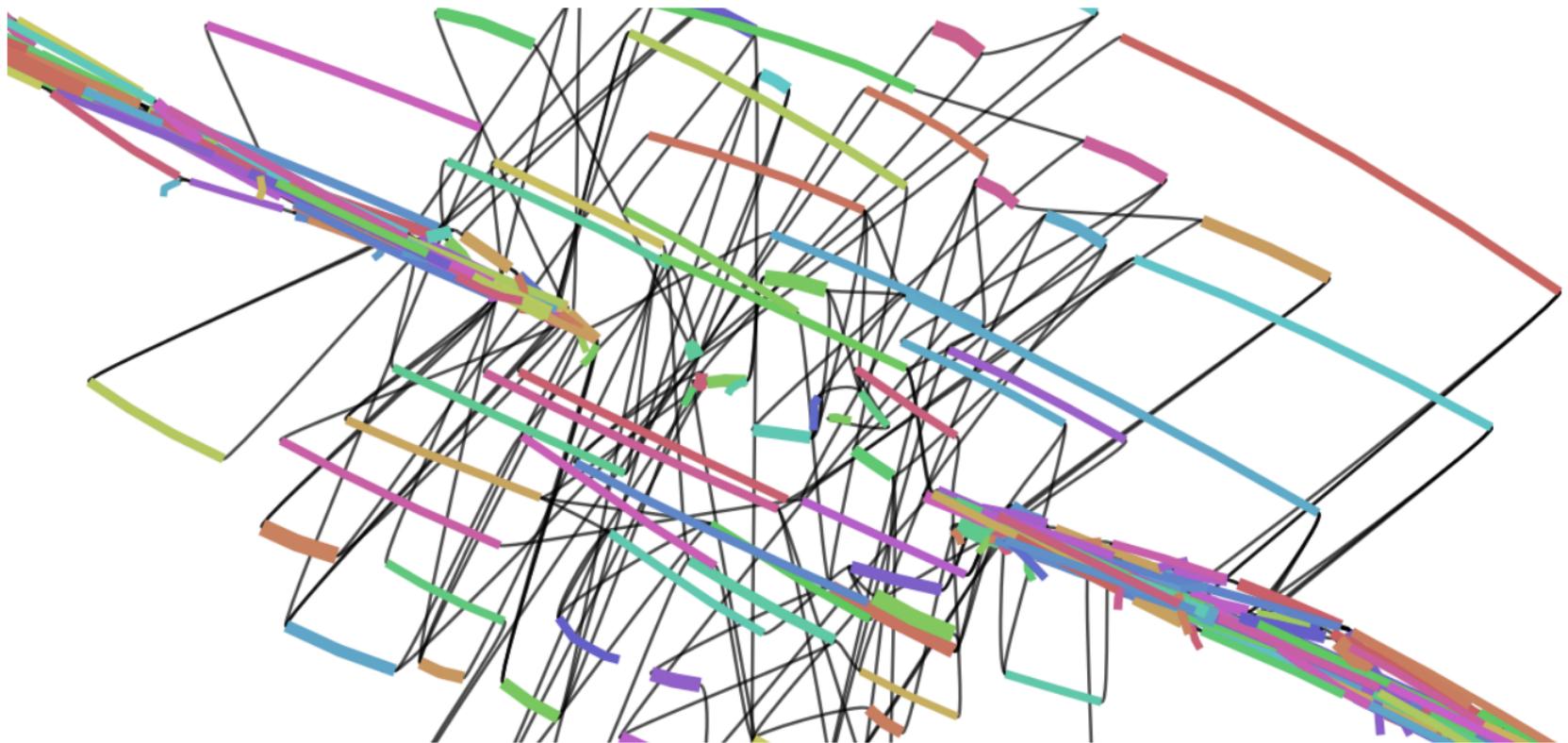
Graphical representation (.gfa plot using Bandage):



- de Bruijn graph on a real dataset



- de Bruijn graph on a real dataset ZOOMED IN



- Sequencing errors

Reads:

ATCGCTATCG  
GGTTTCGTTA  
ATCGATAACGG

TCGCTA  
GGTTTC  
ATCGAT

...

Are not genomic kmers...

- Erroneous  $k$ -mers vs genomic  $k$ -mers

TAAGAAAGCTCTGAATCAACGGACTGCGACA

Reads:

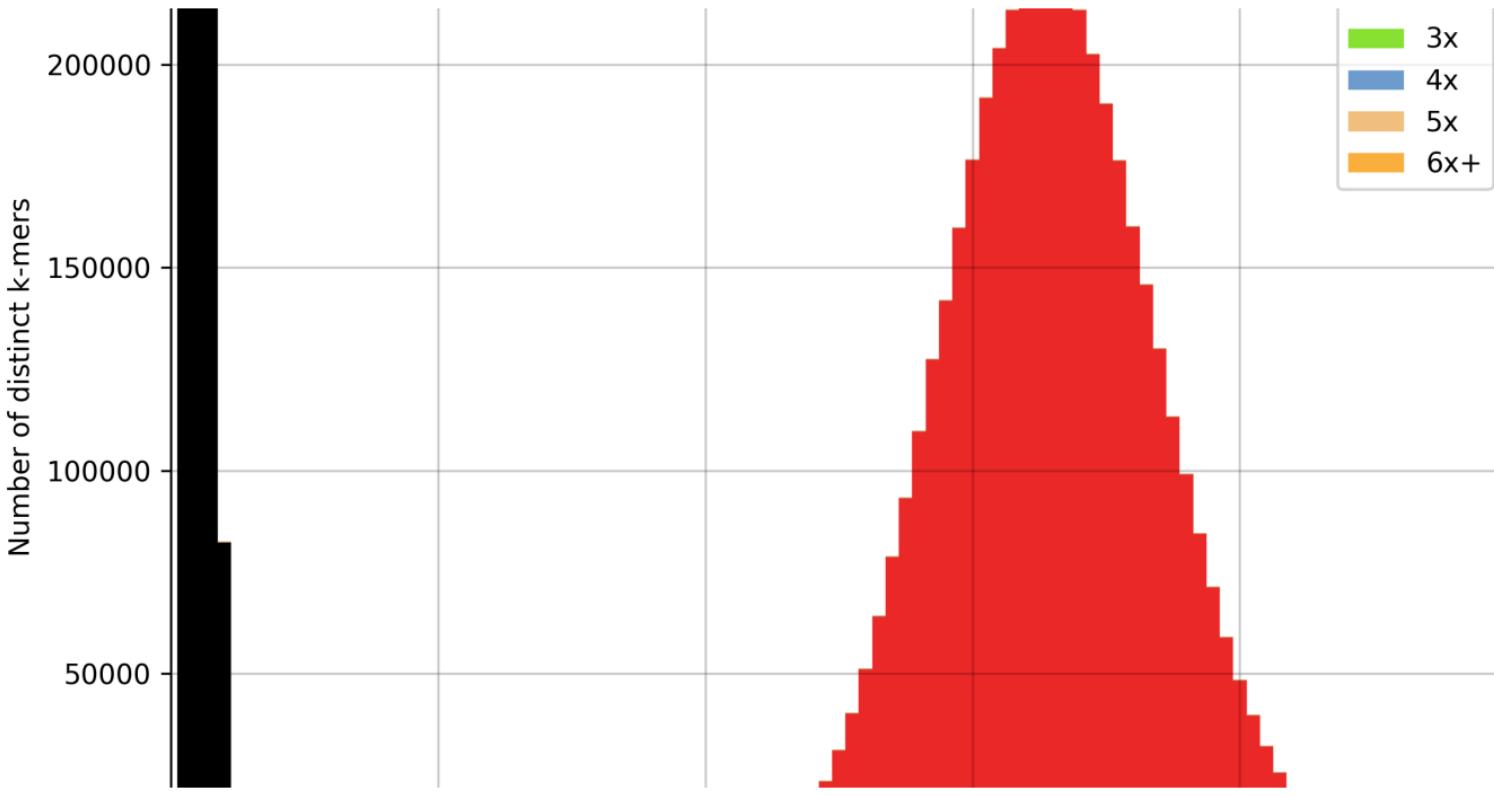
TAAGAAAGCTCTGAATCA  
AAGAAAGCTCTAAATCAAC  
AGAAAGCTCTGAATCAACG  
GAAAGCTCTGAATCAACGGA  
AAAGCTCTGAATCAACGGAC  
AAGCTCTGAATCAACGGACT  
AGCTCTGAATCAACGGACTG  
GCTCTGAATCAACGGCTGC  
CTCTGAATCAACGGACTGCG  
TCTGAATCAACGGACTGCGA

9 times	TCTGAAT
1 time	TCTAAAT
6 times	CAACGGA
1 time	CAACGGT

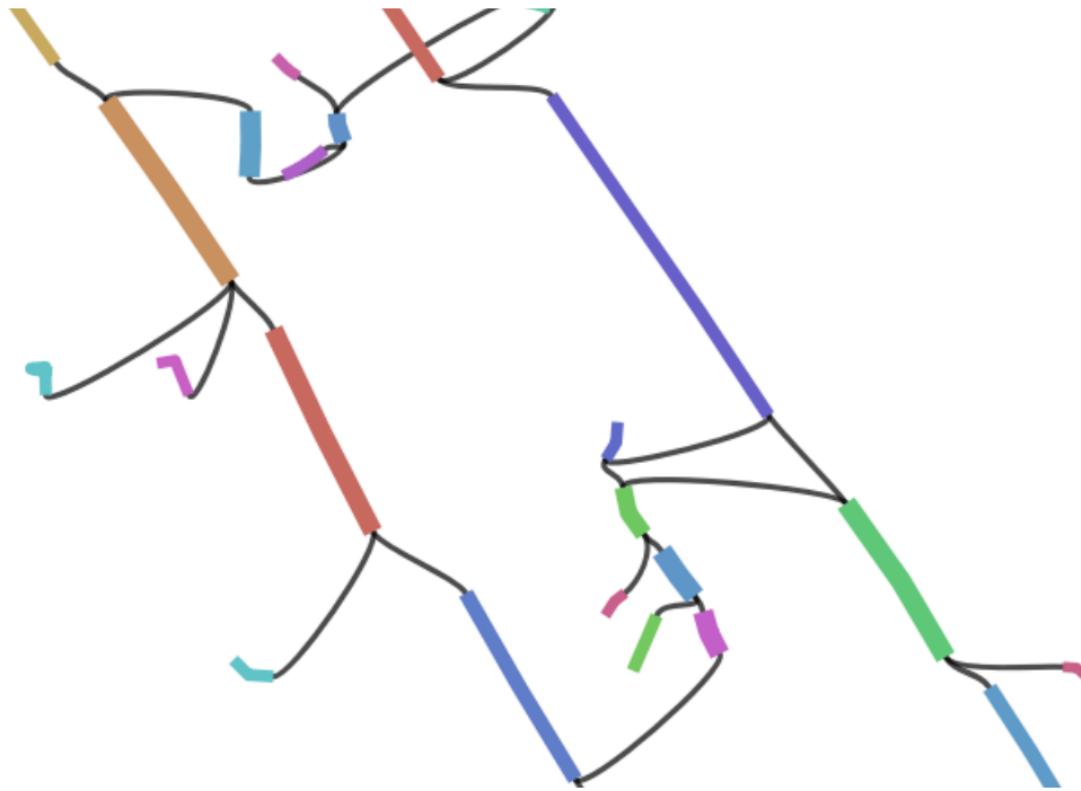
- **Erroneous  $k$ -mers vs genomic  $k$ -mers (ii)**

Erroneous  $k$ -mers are seen less than genomic ones

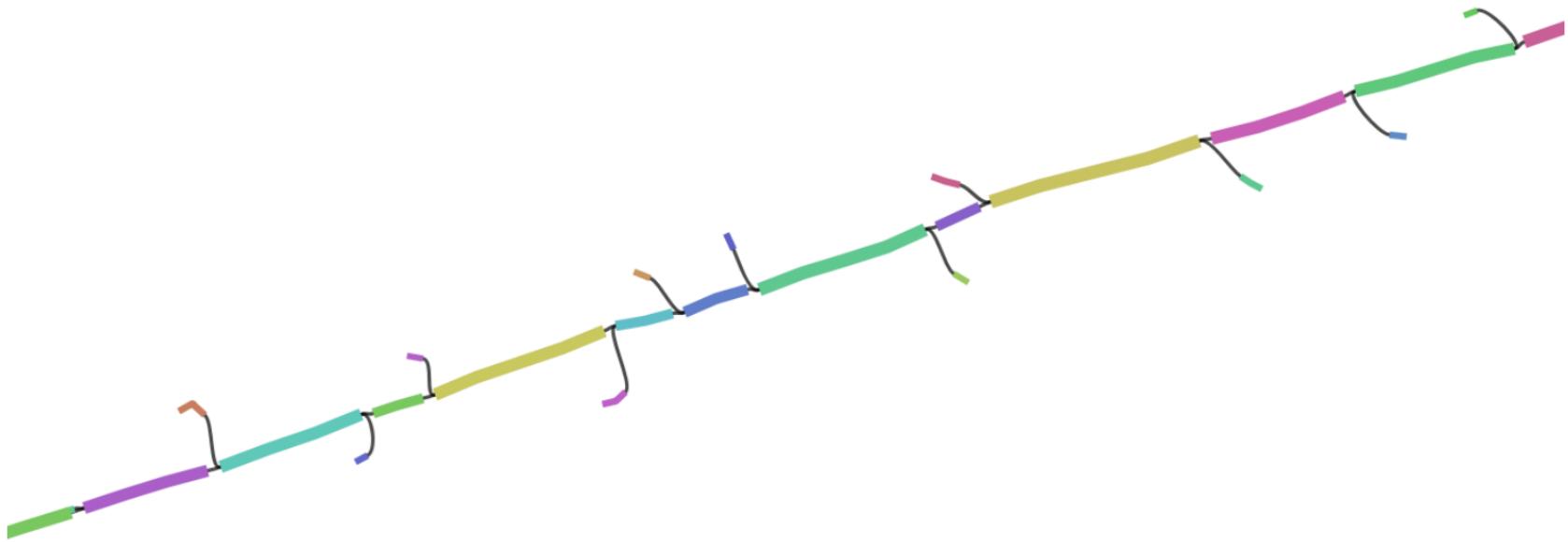
- **K-mer histogram**



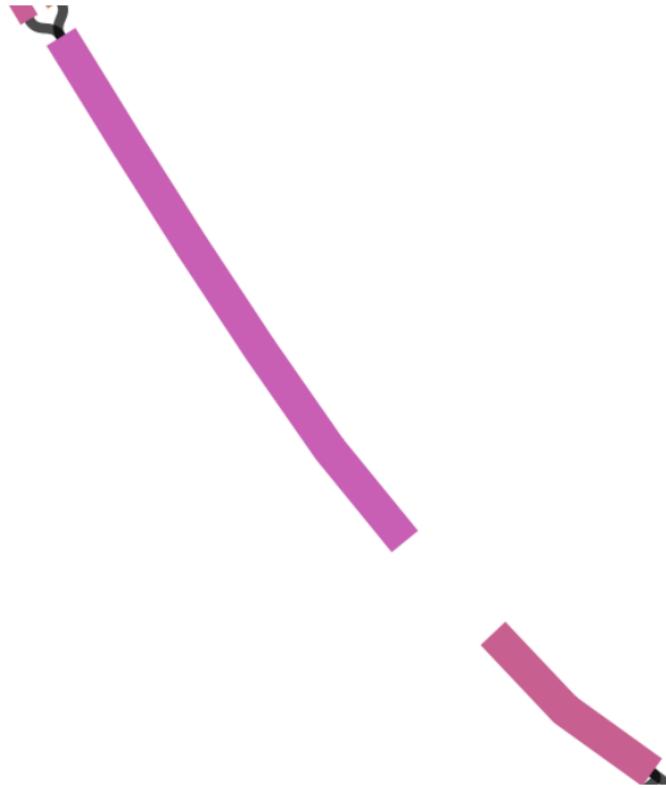
- Removing unique  $k$ -mers



- Removing  $k$ -mers seen less than 3 times



- Removing  $k$ -mers seen less than 4 times



- Errors in de Bruijn graphs

... TACAGGACTTACTGA... genome

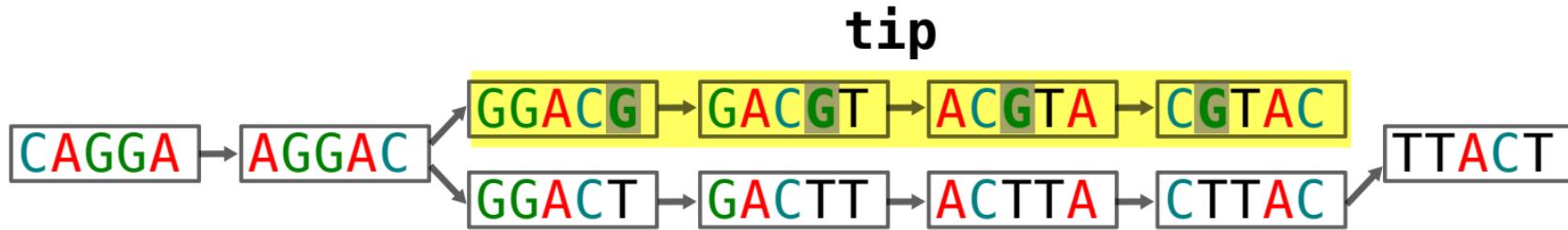
reads CAGGACTTA  
AGGACGTAC ← sequencing error  
AGGACTTAC  
GGACTTACT



- Errors in de Bruijn graphs

... TACAGGACTTACTGA... genome

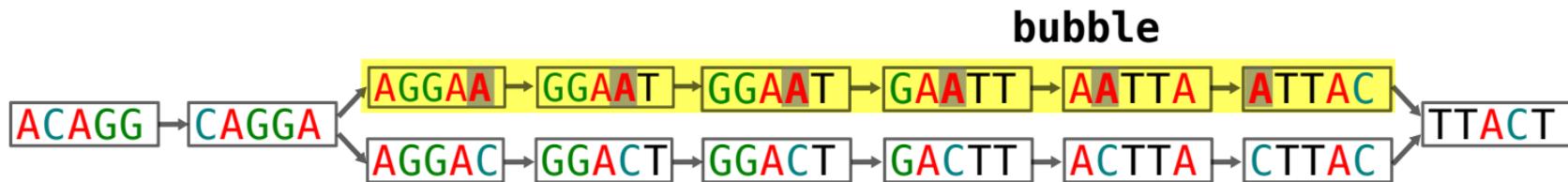
reads CAGGACTTA  
AGGACG**T**TAC ← sequencing error  
AGGAC**T**TAC  
GGAC**T**TACT



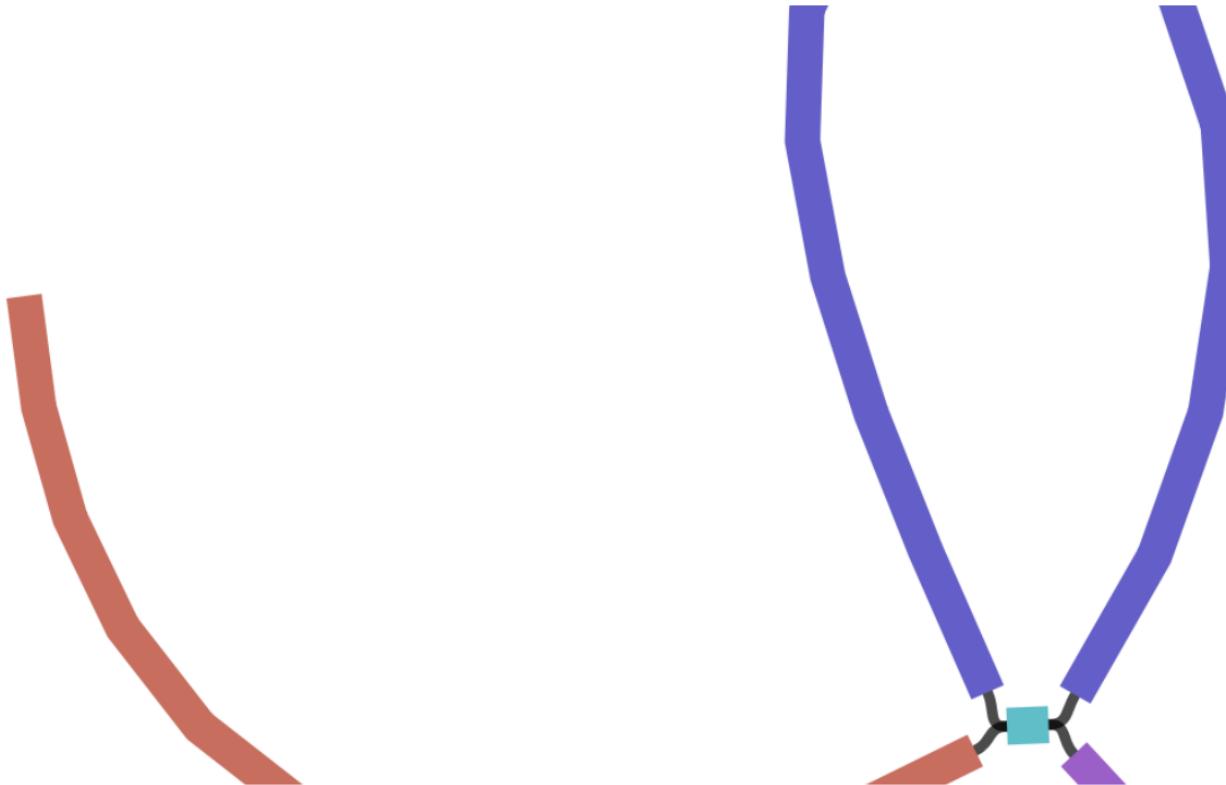
- Errors in de Bruijn graphs

... TACAGGACTTACTGA... genome

reads    ACAGGACTTA  
           CAGGA~~A~~T<sup>T</sup>AC ← **sequencing error**  
           CAGGACTTAC  
           AGGACTTACT



- (Almost assembled phage !)



## • de Bruijn graphs in a nutshell

- Graph of words of size  $k$ ,  $k-1$  overlaps
- Collapses identical  $k$ -mers
- Very successful, have replaced the overlap graphs with high throughput sequencing data
- Still outputs fragments of the genome



White spruce, 20 gigabases

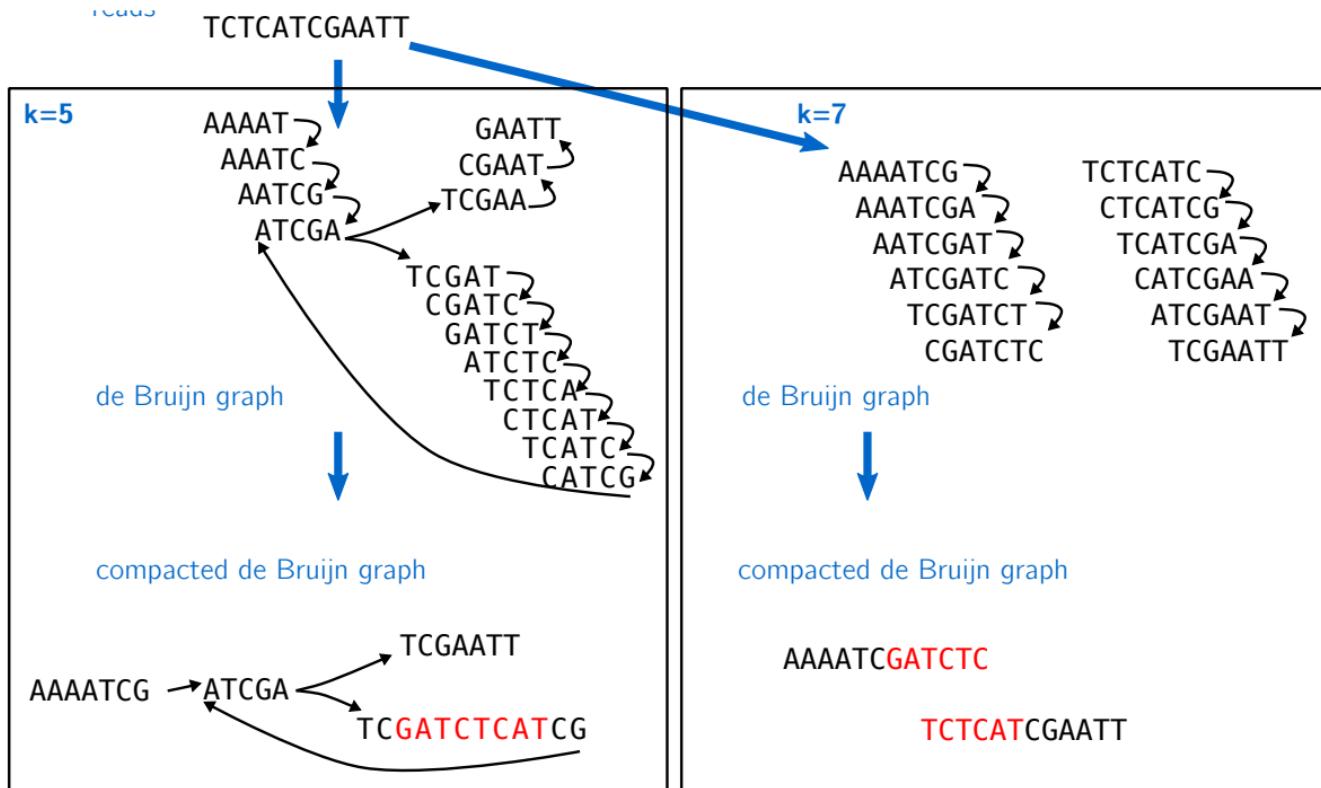
- **Multiple  $k$  assembly**

Most de Bruijn graph assemblers can now perform several assemblies with different  $k$ -mer sizes to produce an improved “super” assembly

**Exercice**

body

## • Multiple $k$ assembly



- **Multiple  $k$  assembly (ii)**

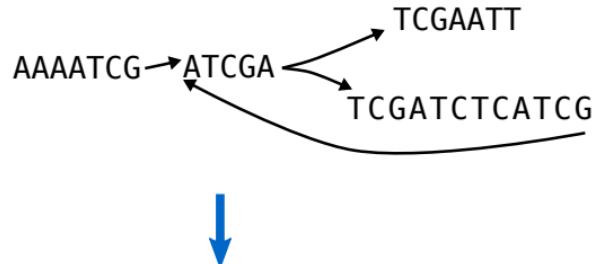
We are missing GATCTCA and ATCTCAT in the second graph.

But they are present in the first graph!

## • Multiple $k$ assembly

compacted de Bruijn graph

$k=5$



k-mers  $k=7$   
from cdBG  $k=5$

AAAATCG

AAATCGA

TCGATCT  
CGATCTC  
GATCTCA  
ATCTCAT  
TCTCATC  
CTCATCG



k-mers  $k=7$   
from the reads

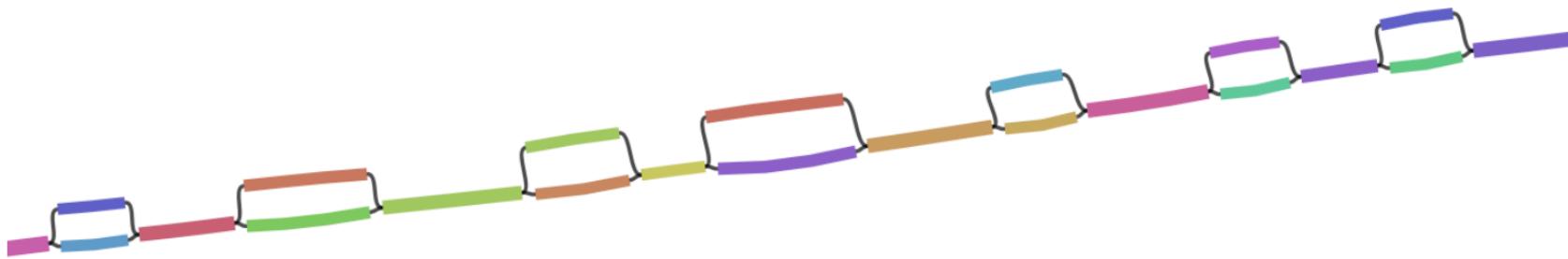
AAAATCG TCTCATC  
AAATCGA CTCATCG  
AATCGAT TCATCGA  
ATCGATC CATCGAA  
TCGATCT ATCGAAT  
CGATCTC TCGAATT

compacted de Bruijn graph  
 $k=7$



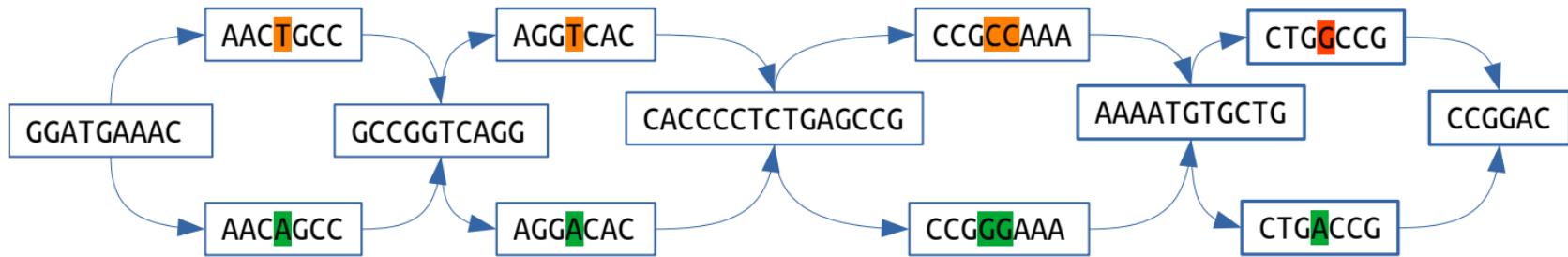
AAAATCGATCTCATCGAATT

- de Bruijn graph on an eukaryota



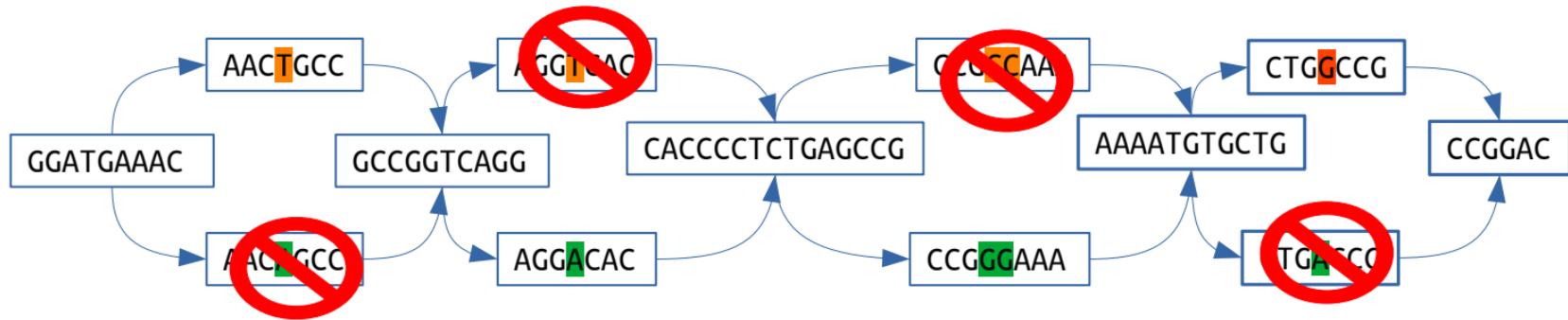
- Two or more genomes per individual

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC



- Two or more genomes per individual

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC



Assembly:

- Assembly concession number 2: collapse variability

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC

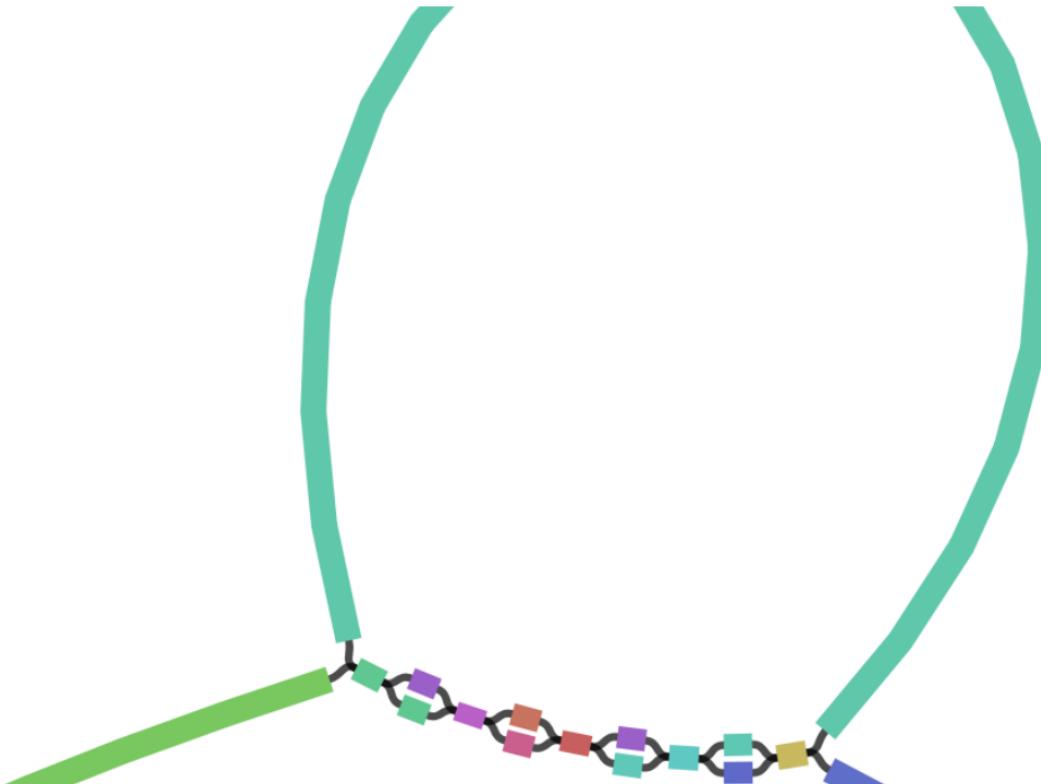
Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGCCGGAC

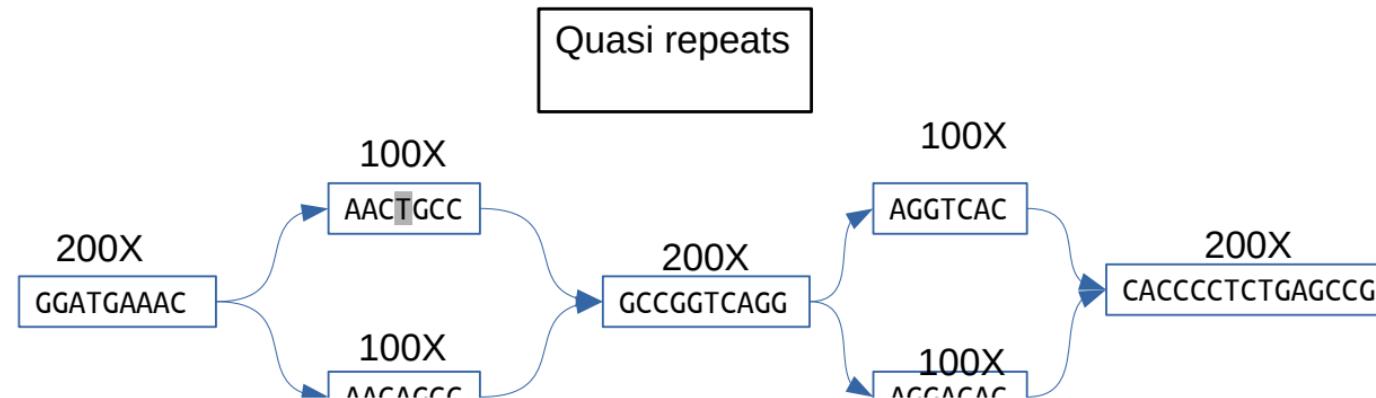
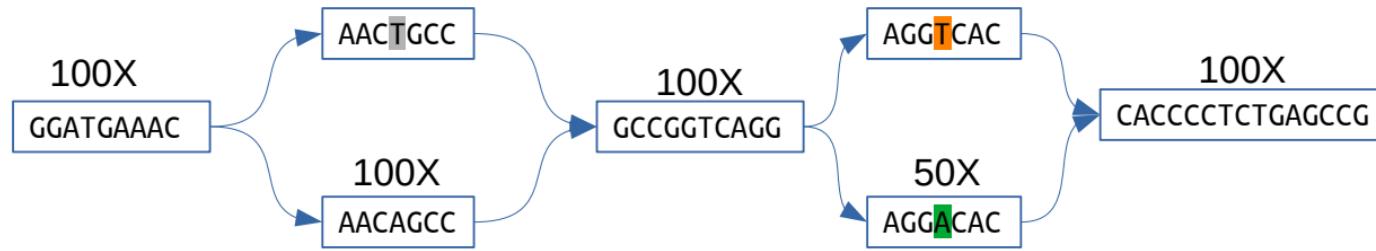
Reads:

GATGAAACTG  
ATGAAACAGC  
TGAAACAGCCG  
GAAACTGCCGG  
AAACTGCCGGT  
AACAGCCGGTC  
ACAGCCGGTCA  
CTCCCCCTCAAC

- Paralog genes/repeats



- Paralog genes/repeats in graph



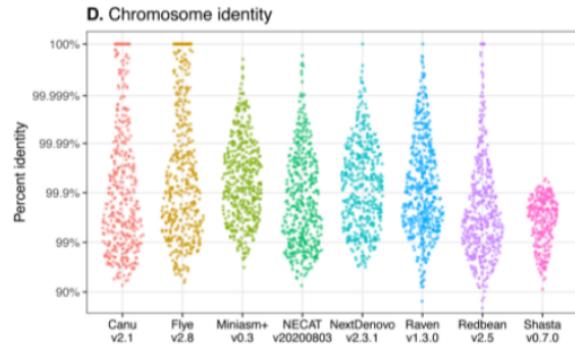
- An assembler is a set of heuristics

### Graph cleaning heuristics

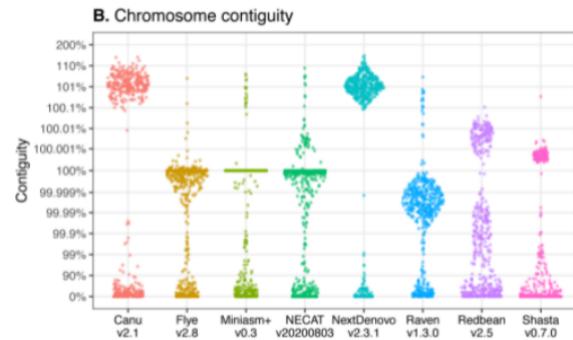
body

- An assembly is a model

1. Assemblies contain errors
2. Different tools can produce very similar assemblies
3. A single tool can produce very different assemblies with small changes of parameters(!)



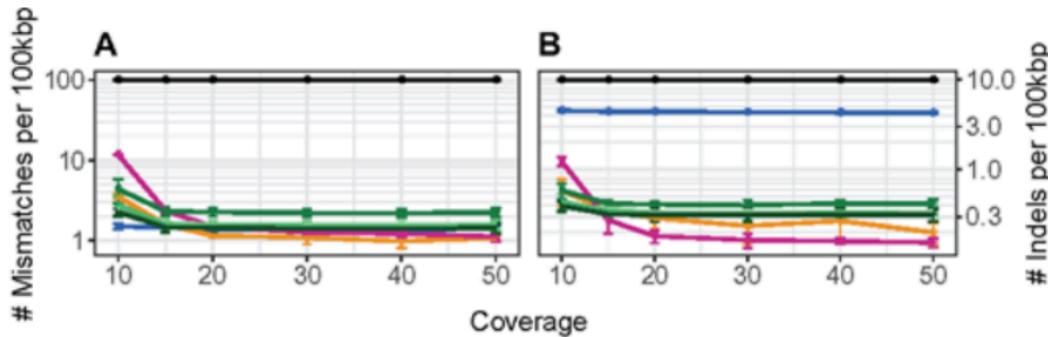
- An assembly is a model (ii)



From <https://github.com/rrwick/Long-read-assembler-comparison>

- What do we do post-assembly?

1. Assess its quality
2. Improve it
3. Use it!

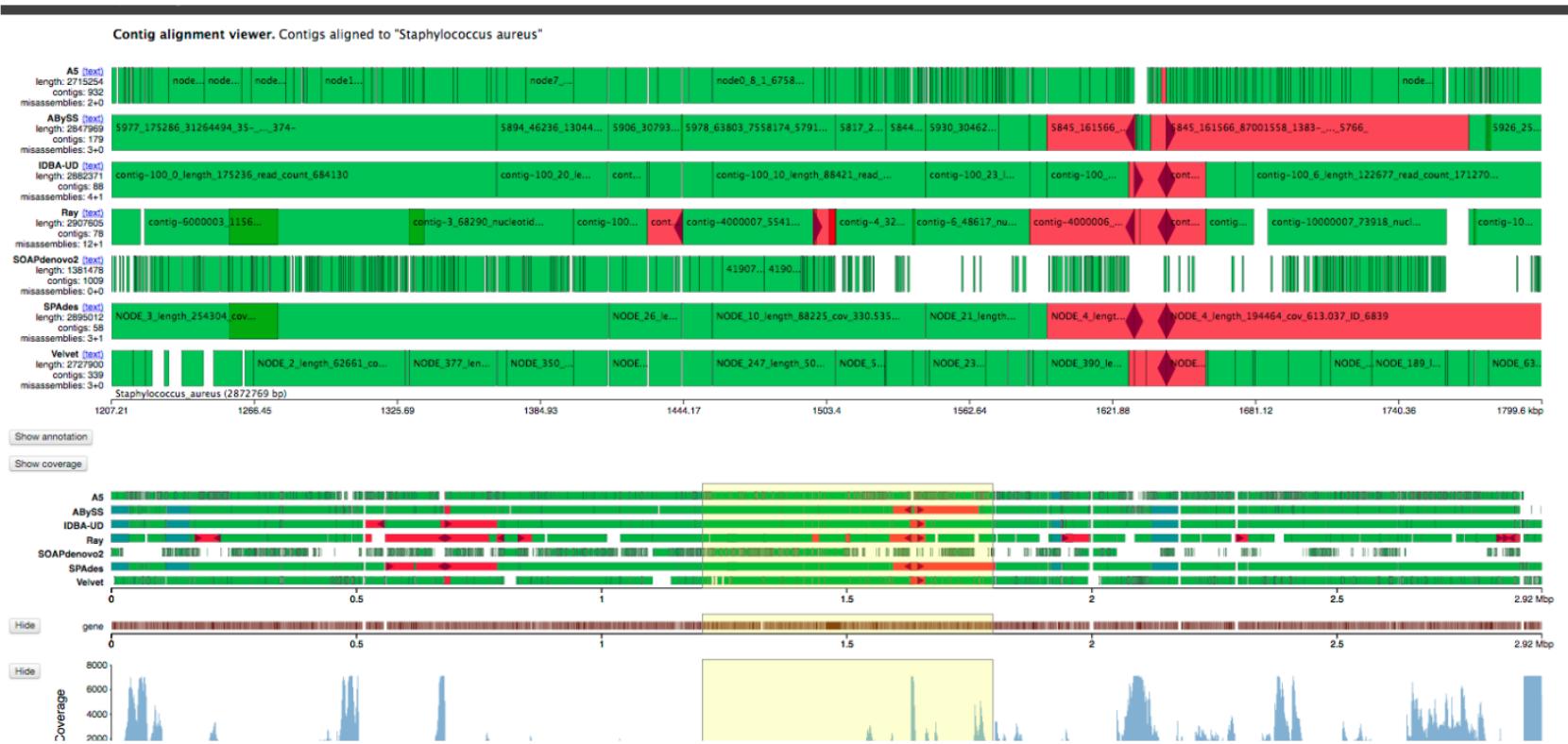


Two ways to polish an assembly according to MidJourney

- Evaluate assembly according to a reference

Contigs can be mapped and compared to a reference/closely related genome

## • Evaluate assembly according to a reference (ii)



- Evaluate assembly according to a reference (iii)

From <https://quast.bioinf.spbau.ru/manual.html>

## • Assembly statistics

Total aligned length	4 776 214	4 568 317	4 553 809	4 550 150
NGA50	69 801	122 647	133 309	112 446
LGA50	21	14	12	14

### Misassemblies

# misassemblies	4	0	0	4
Misassembled contigs length	231 767	0	0	435 515

### Per base quality

# mismatches per 100 kbp	2.09	2.69	1.03	3.19
# indels per 100 kbp	0.57	1.31	0.29	1.98
# N's per 100 kbp	24.59	0	17.55	94.19

### Statistics without reference

# contigs	176	95	92	90
Largest contig	248 481	235 933	285 196	264 944
Total length	4 777 853	4 571 292	4 557 363	4 552 266
Total length (>= 1000 bp)	4 757 929	4 562 458	4 548 710	4 544 453
Total length (>= 10000 bp)	4 562 801	4 478 614	4 466 223	4 475 223
Total length (>= 50000 bp)	3 248 113	3 833 793	3 812 315	3 817 904

### BUSCO completeness

- **Assembly statistics (ii)**

From <http://cab.cc.spbu.ru/quast/>

- Assembly continuity

N50

body

Example: 1 Mbp genome

50%



- **Assembly continuity**

N50

body

N75

body

NGA50

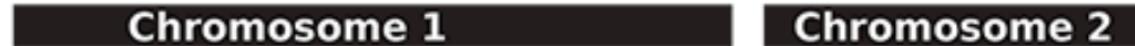
body

- Misassemblies

**Contig**



**Reference**



**Relocation**



**Inversion**



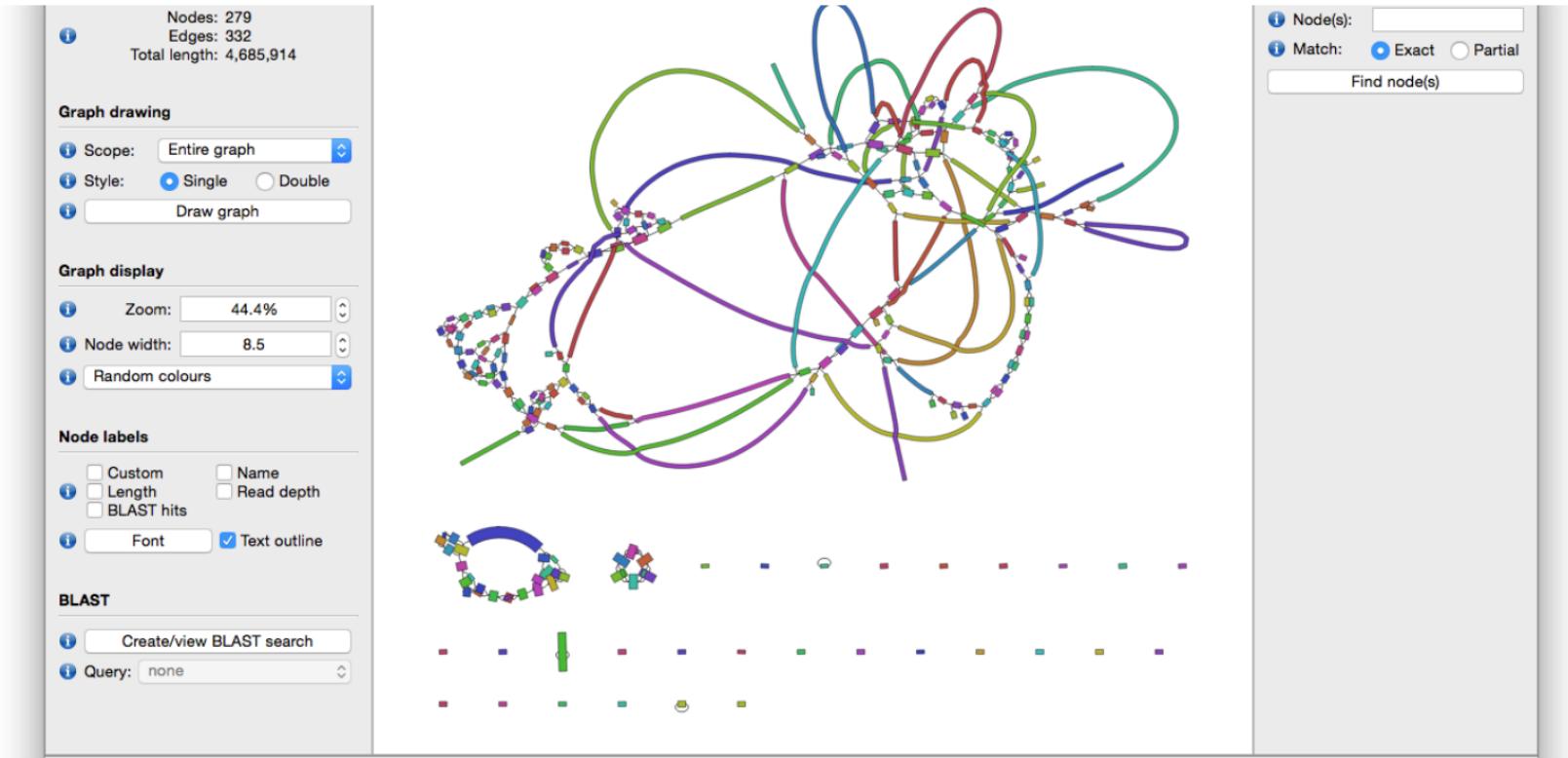
**Translocation**



- **Visualize assembly**

Bandage tool can visualize assembly graphs (GFA)

## • Visualize assembly (ii)

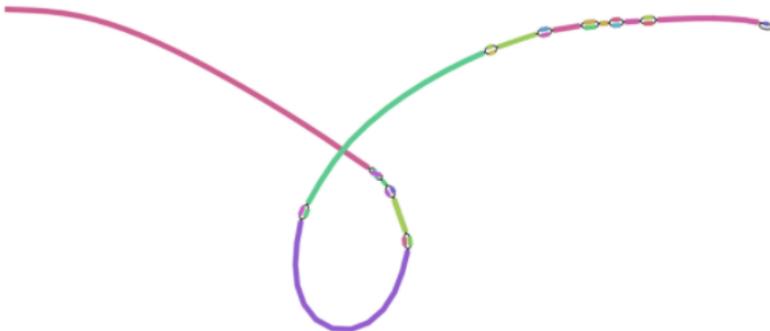


- **Visualize assembly (iii)**

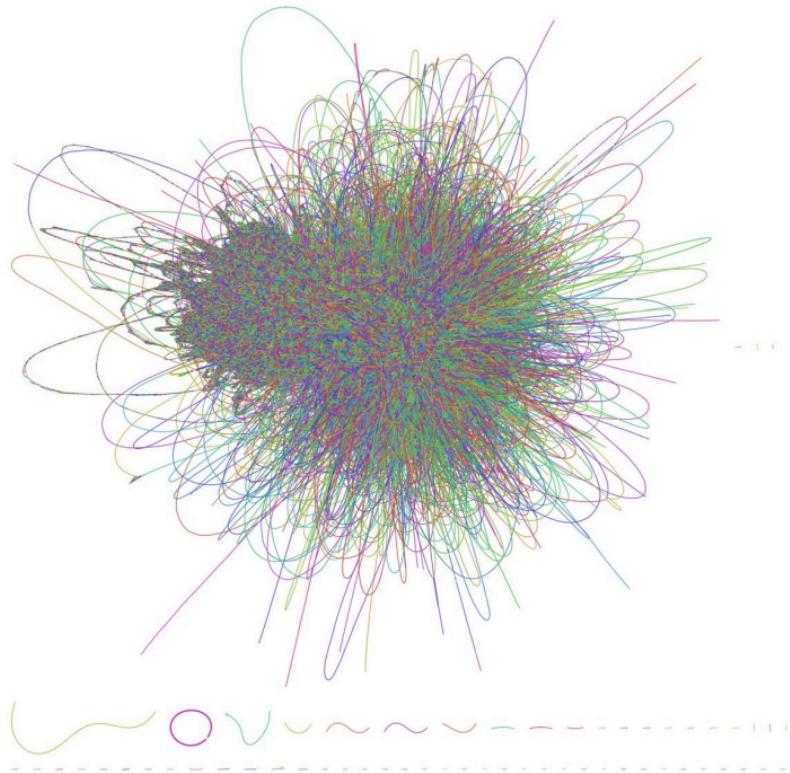
From <https://rwick.github.io/Bandage>

- **Visualize assembly**

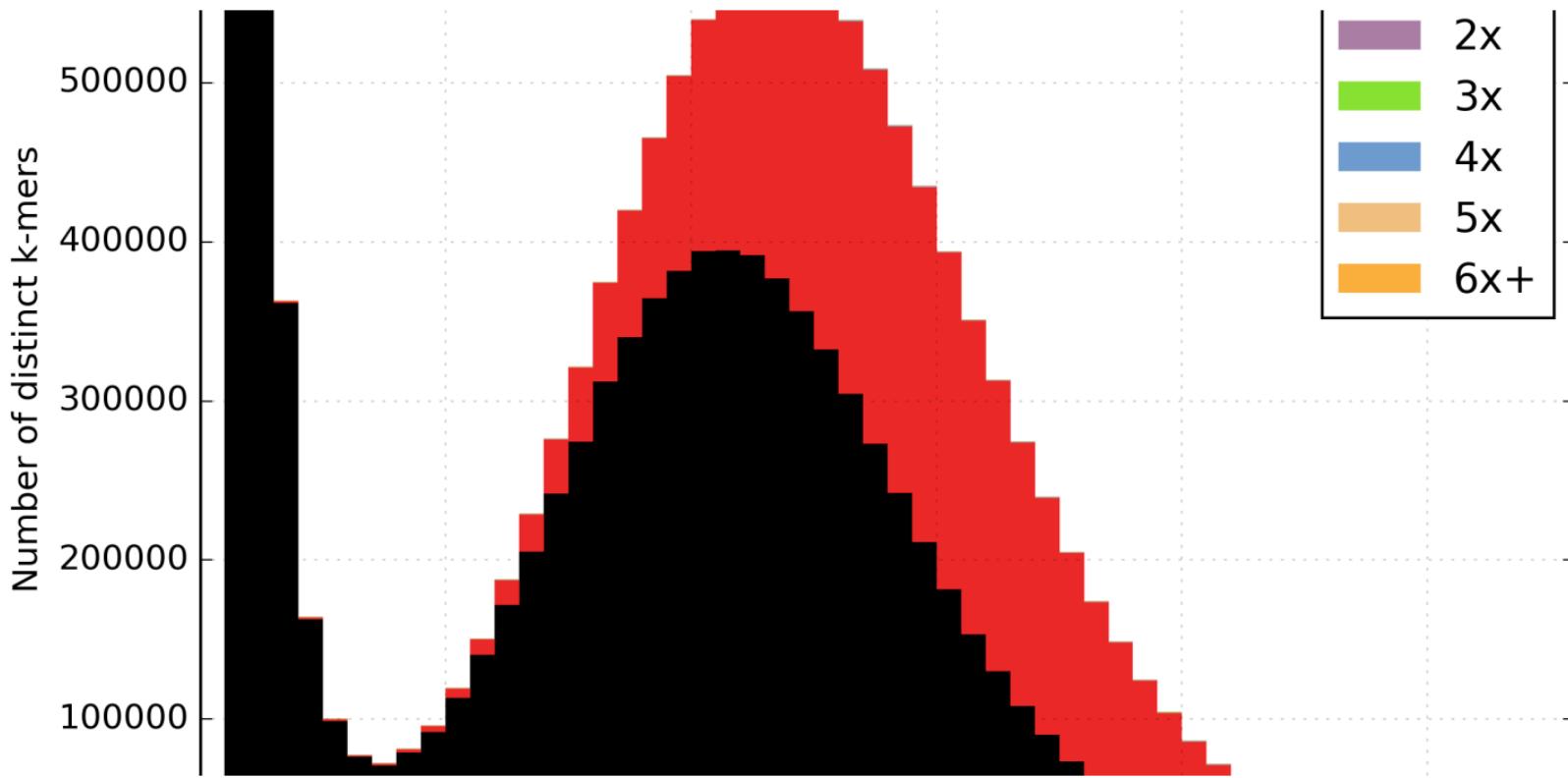
Bandage tool can visualize assembly graphs (GFA)



- Visualize assembly (ii)



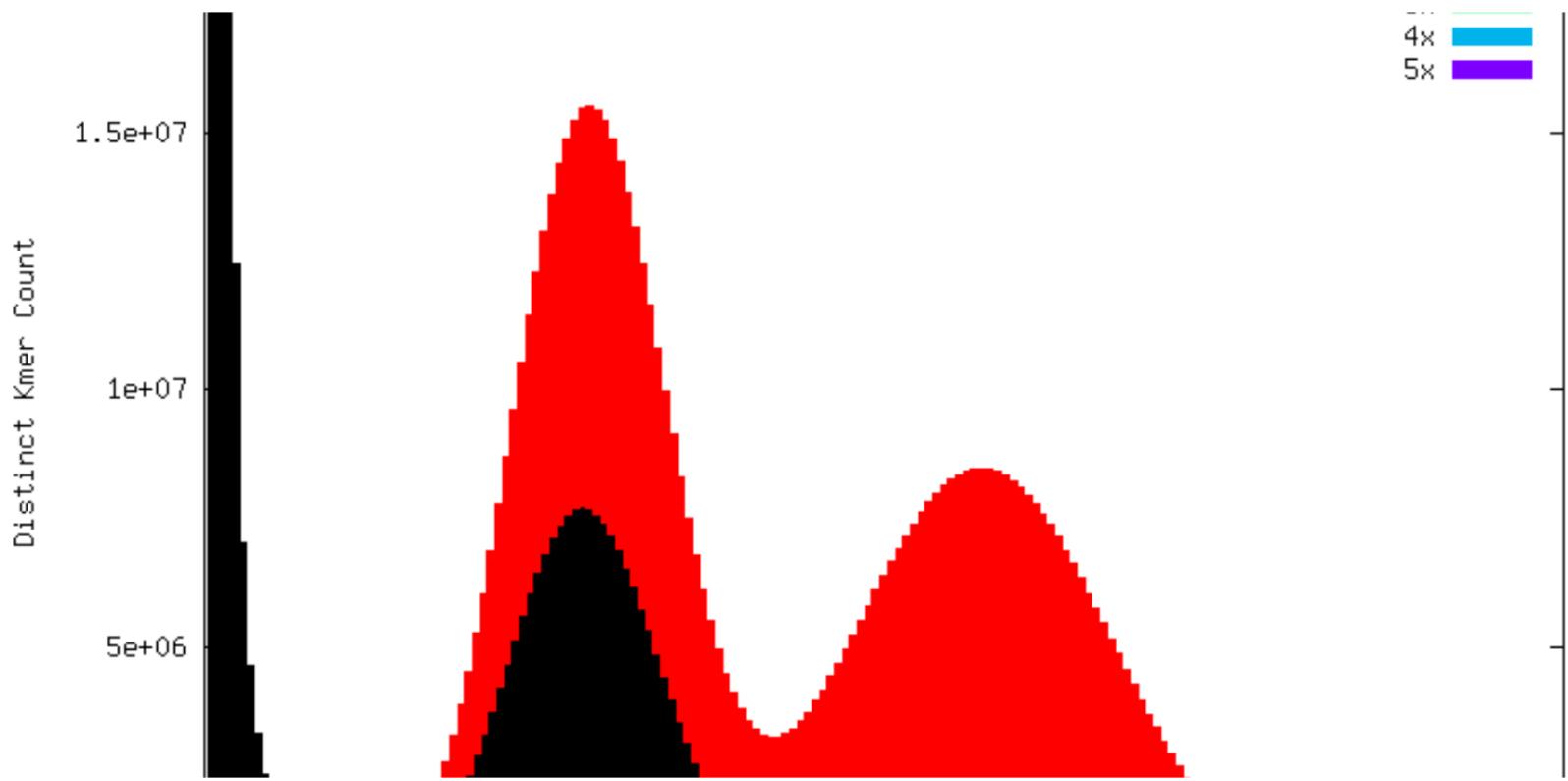
- **K-mer spectrum visualization with KAT**



- ***K*-mer spectrum visualization with KAT (ii)**

From <https://kat.readthedocs.io/en/latest/>

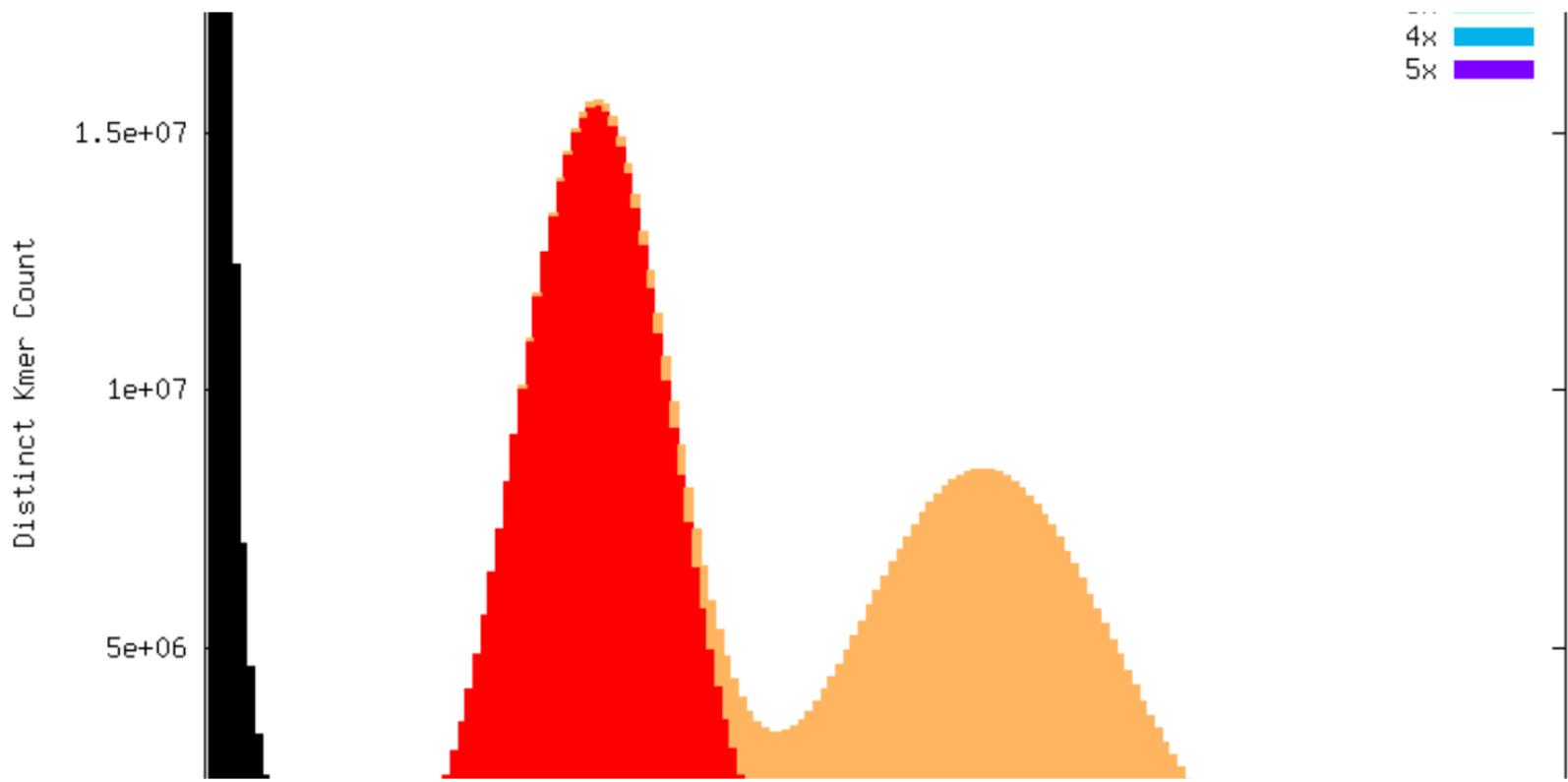
- *K*-mer spectrum visualization with KAT



- ***K*-mer spectrum visualization with KAT (ii)**

From <https://kat.readthedocs.io/en/latest/>

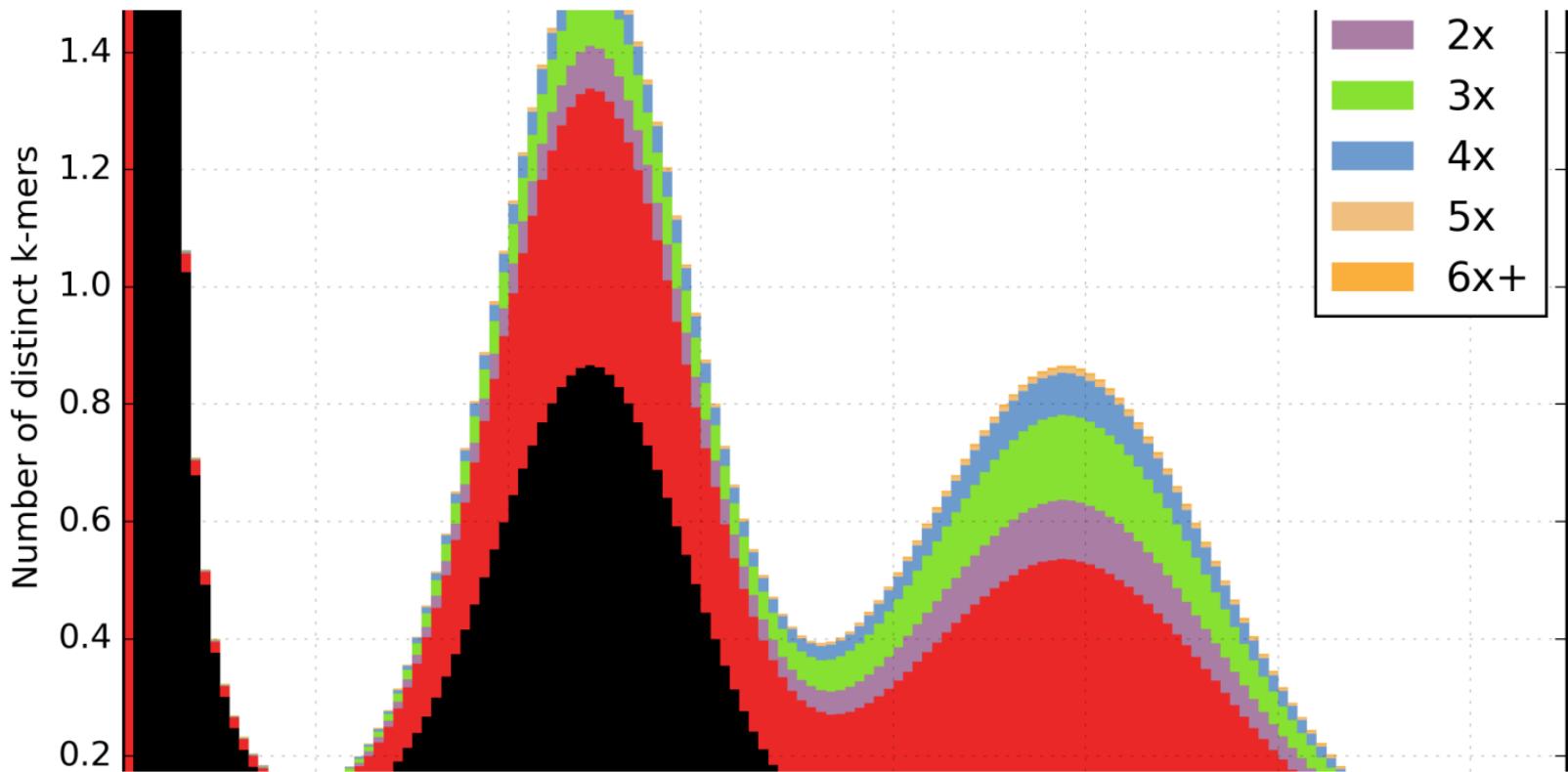
- *K*-mer spectrum visualization with KAT



- ***K*-mer spectrum visualization with KAT (ii)**

From <https://kat.readthedocs.io/en/latest/>

- *K*-mer spectrum visualization with KAT



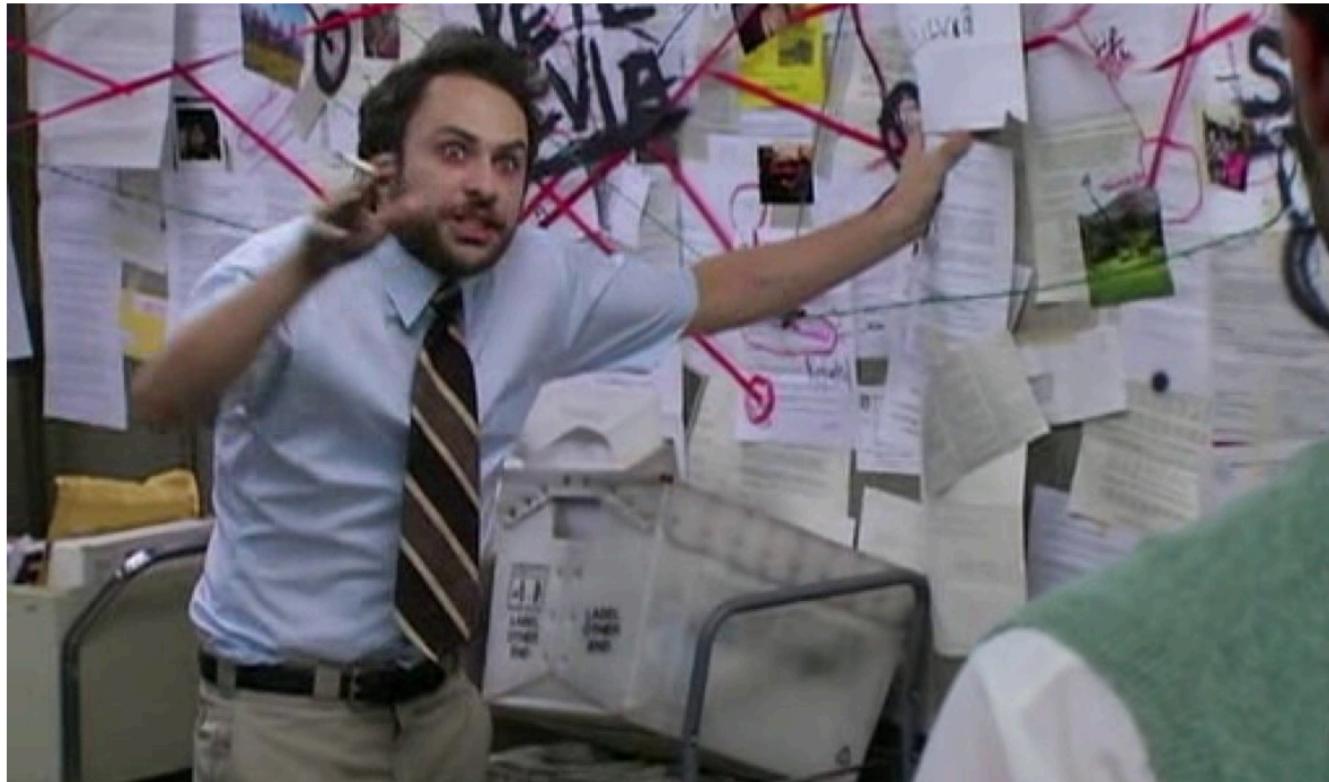
- ***K*-mer spectrum visualization with KAT (ii)**

From <https://kat.readthedocs.io/en/latest/>

- **The end**

... of the theoretical part (or is it?)

- The end (ii)



- Sanger



- Medium reads approx 1000 bp
- Very low error rate approx 0.01 %
- Low throughput (up to billion of reads per run)
- Costly (\$500/Mb)

- **Sanger (ii)**

No longer used for assembly

- Second generation sequencing



NextSeq Series



HiSeq 4000 System



HiSeq X Series‡

NovaSeq 6000  
System

- Short reads approx 150 bp
- Low error rate < 1%
- High throughput (up to billion of reads per run)
- Cheap (\$0.50/Mb)
- GC bias

- Which assembly strategy is best suited?

- Short reads approx 100 bp
- Low error rate below 1%
- High throughput (up to billions of reads per run) Based on long reads properties, which assembly solution would you choose and why?

Vote!

- Greedy
- Overlap graph
- de Bruijn graph

- Paradigm Breakdown

Greedy

body

Overlap graph

body

De Bruijn graph

body

- State-of-the-art

Well performing assemblers

body



- State-of-the-art (ii)



- State-of-the-art (iii)

**Other notable assemblers**

body

- Third generation sequencing



- Third generation sequencing (ii)



- Long reads approx 10-100 kbp
- High error rate (up to

10%

)

- **Third generation sequencing (iii)**

- High throughput (up to millions of reads per run)

- Nanopore VS Pacbio

Nanopore

body

Pacbio

body

- Repeats spanning

GGTAATG TTTTTT GTGCTAAT GTTTTTT ATGGATG TTTTTTA  
ATGGTTT AATGCCGT ATGTCGT TTTATCTG  
TTTGGTG TTTTCATG CGTAATT

Contexts of the repeat:

...ATGG

ATCT...

...TGCG

???TTTTTT???

GGTG...

CATG...

- Repeats spanning

Reads:

The diagram shows several short DNA reads represented as horizontal lines with colored segments. The colors represent different nucleotides: yellow for A, red for T, cyan for G, and grey for C. The reads are: GGTAATG, TTTTTT, GTGCTAAT, GTTTTTT, ATGGATG, TTTTTTA, ATGGTTT, AATGCCGT, ATGTCTG, TTTATCTG, TTTGGTG, TTTTCATG, CTGAATT, and CTGAATTT.

Long reads:

The diagram shows three long DNA reads represented as horizontal lines with colored segments. The colors represent different nucleotides: yellow for A, red for T, cyan for G, and grey for C. The reads are: TGGTTTTTTGGT, TGCCTTTTTCAT, and TGAATTTTTATCT.

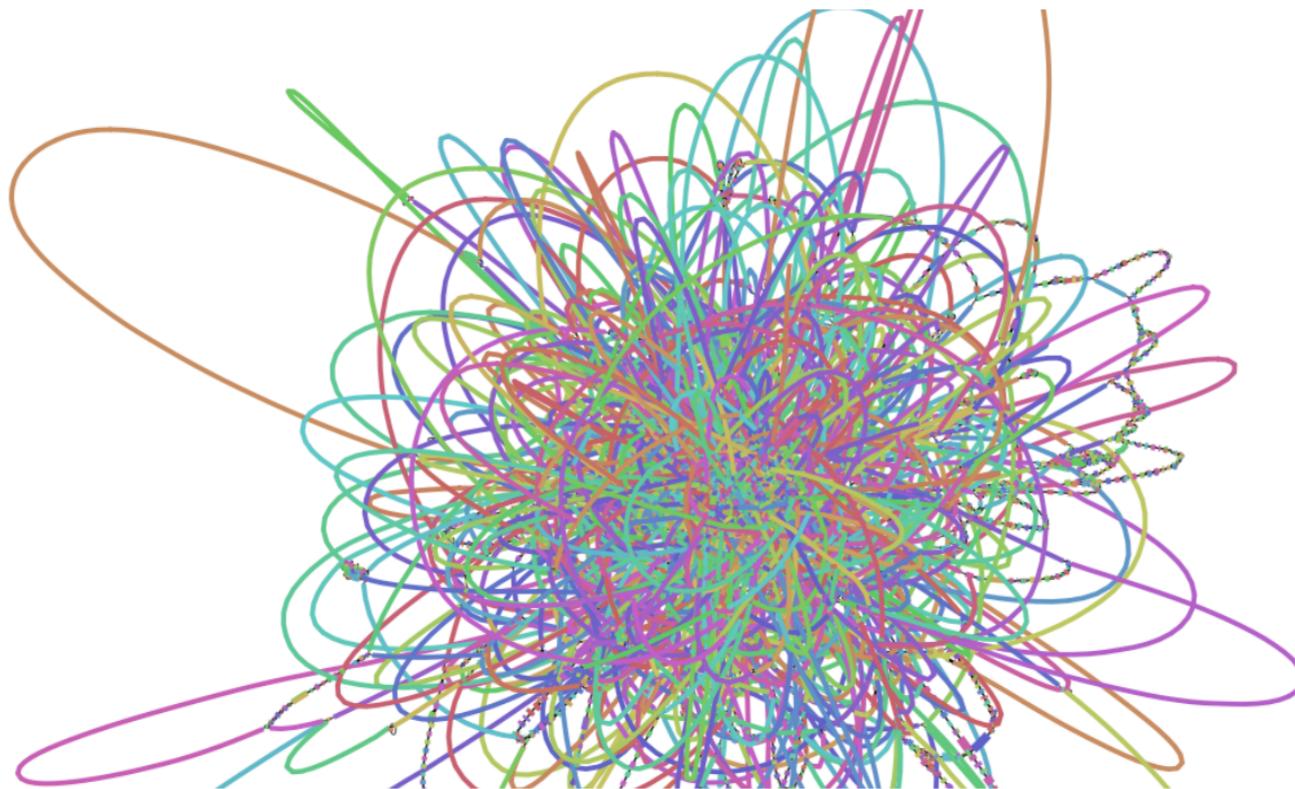
Contexts of the repeats:

The diagram shows three contexts of repeat sequences with green arrows indicating transitions. The contexts are: ...ATGG..., ...TGCG..., and ...GTAA... .

- Read length matters

Read size=21

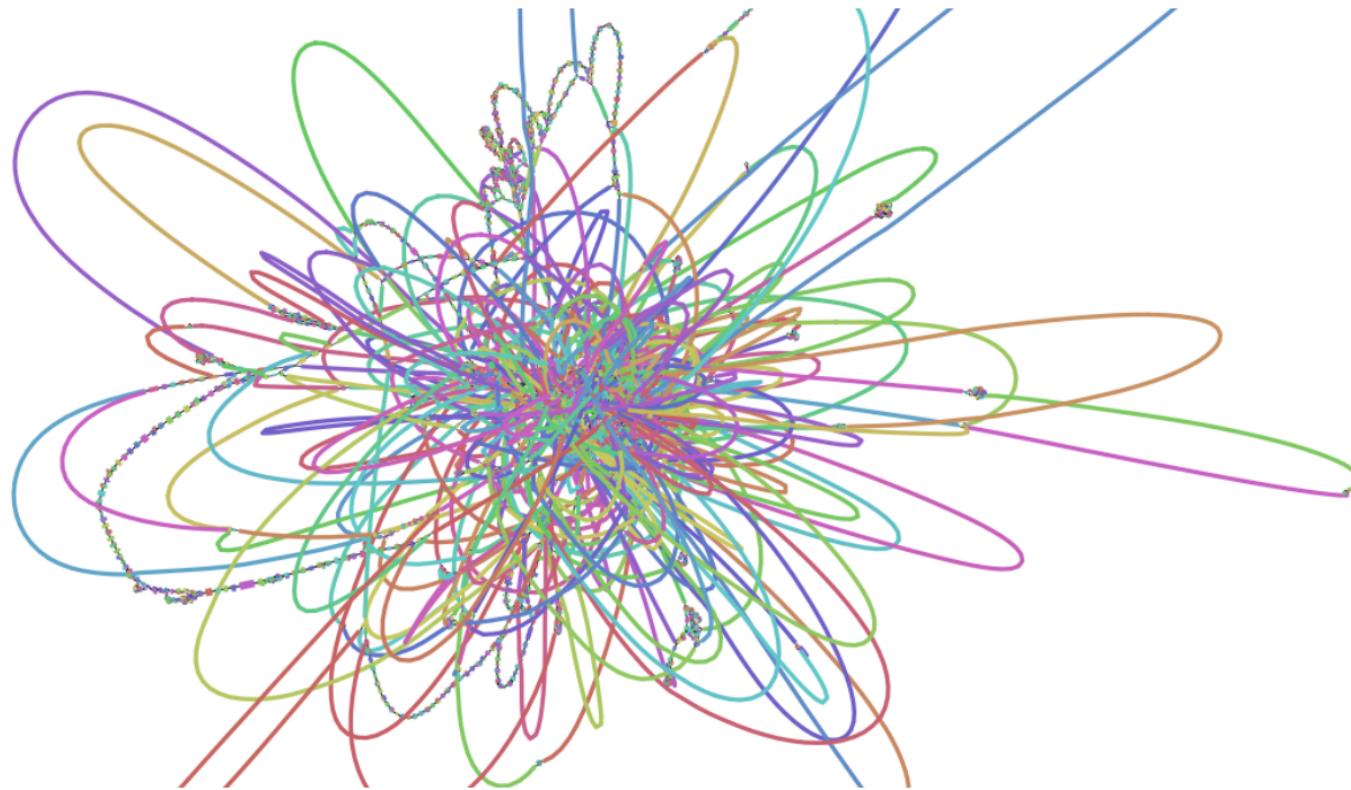
- Read length matters (ii)



- Read length matters

Read size=31

- Read length matters (ii)



- Read length matters

Read size=63

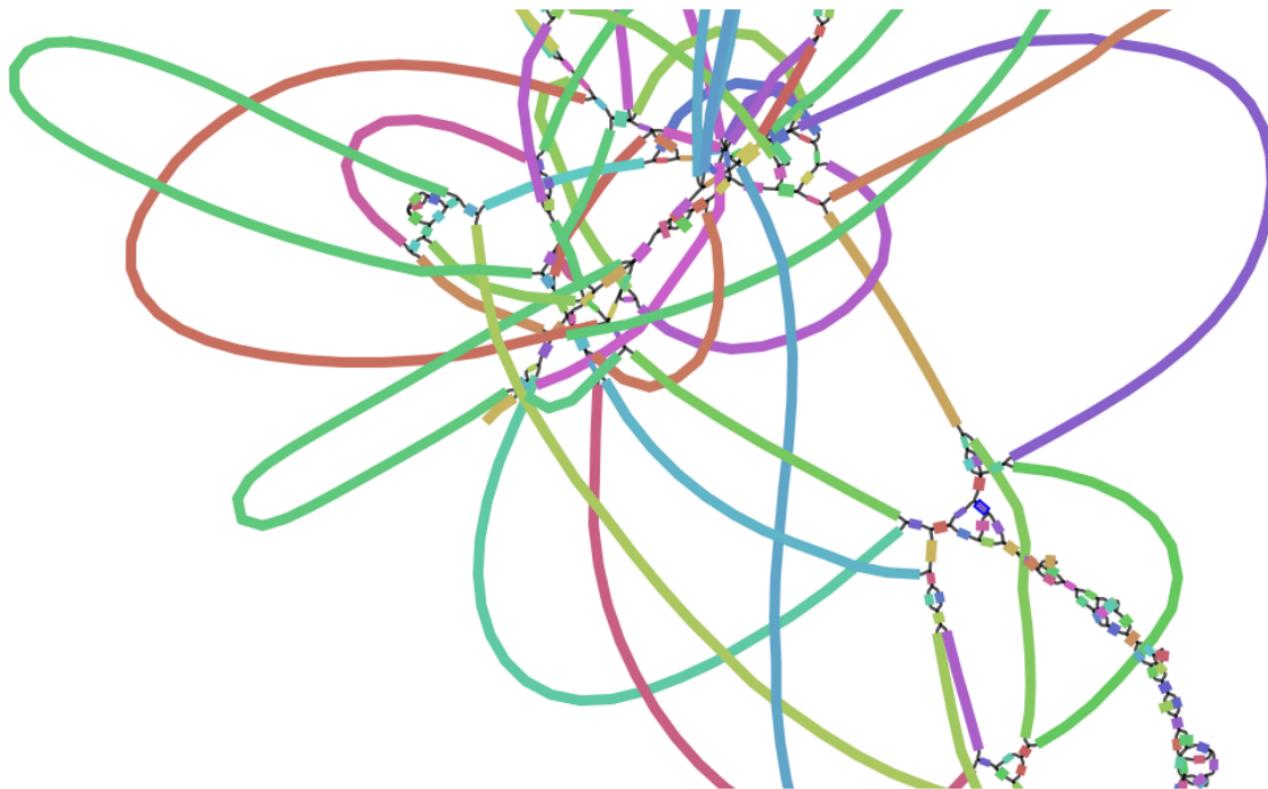
- Read length matters (ii)



- Read length matters

Read size=255

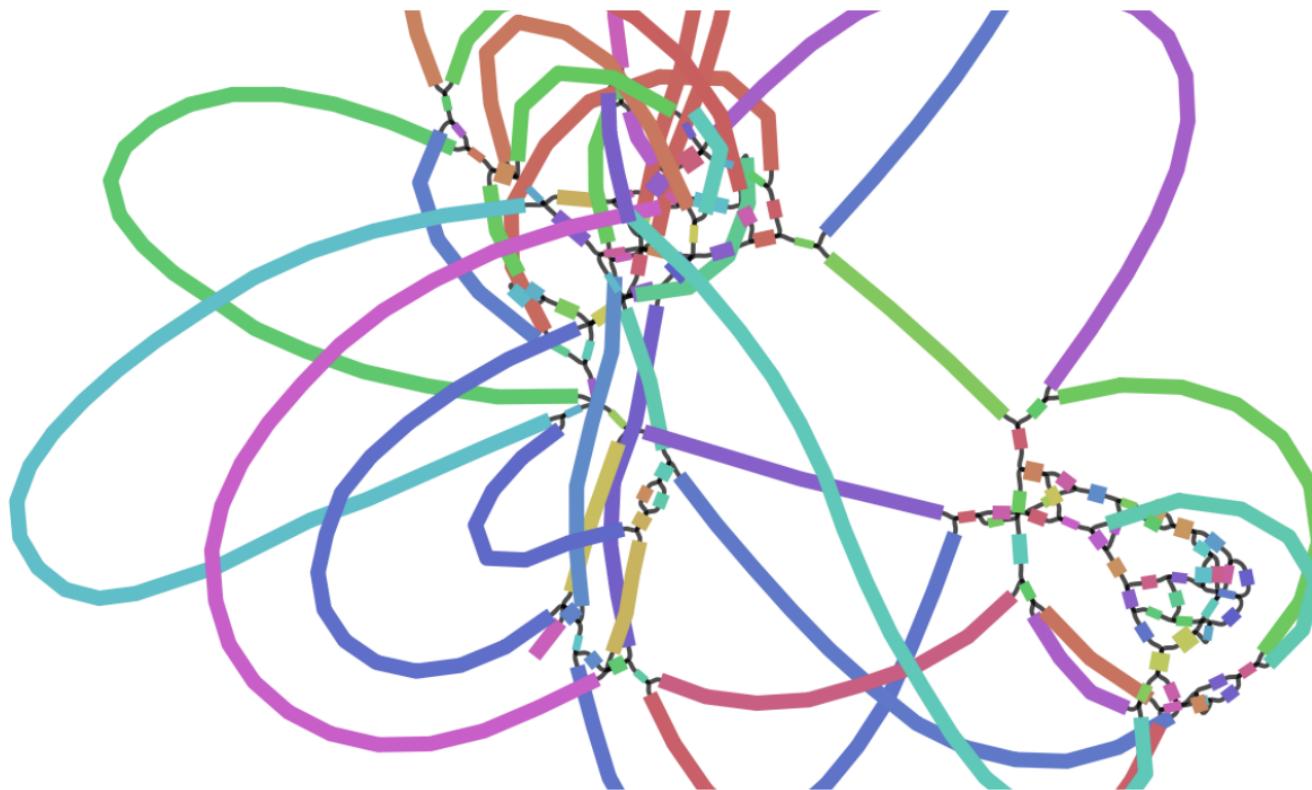
- Read length matters (ii)



- **Read length matters**

Read size=500

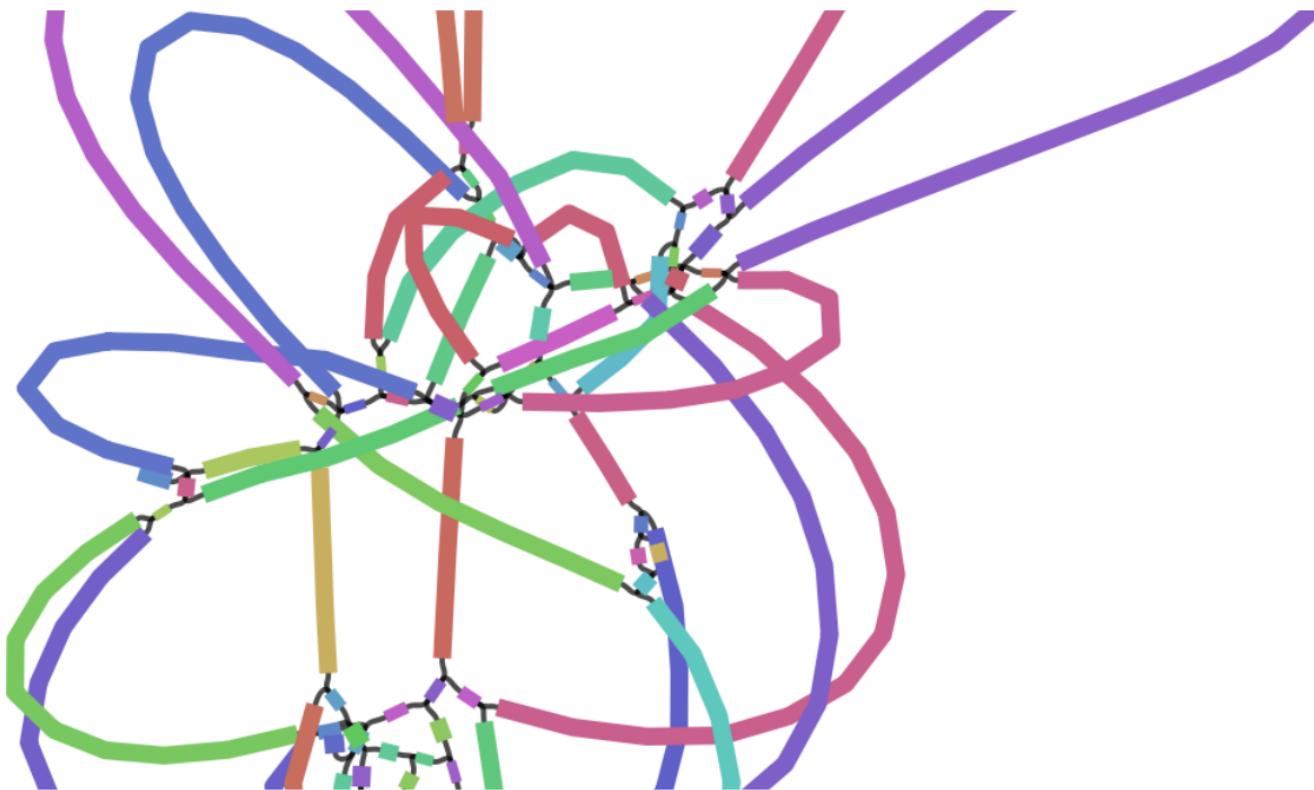
- Read length matters (ii)



- **Read length matters**

Read size=1000

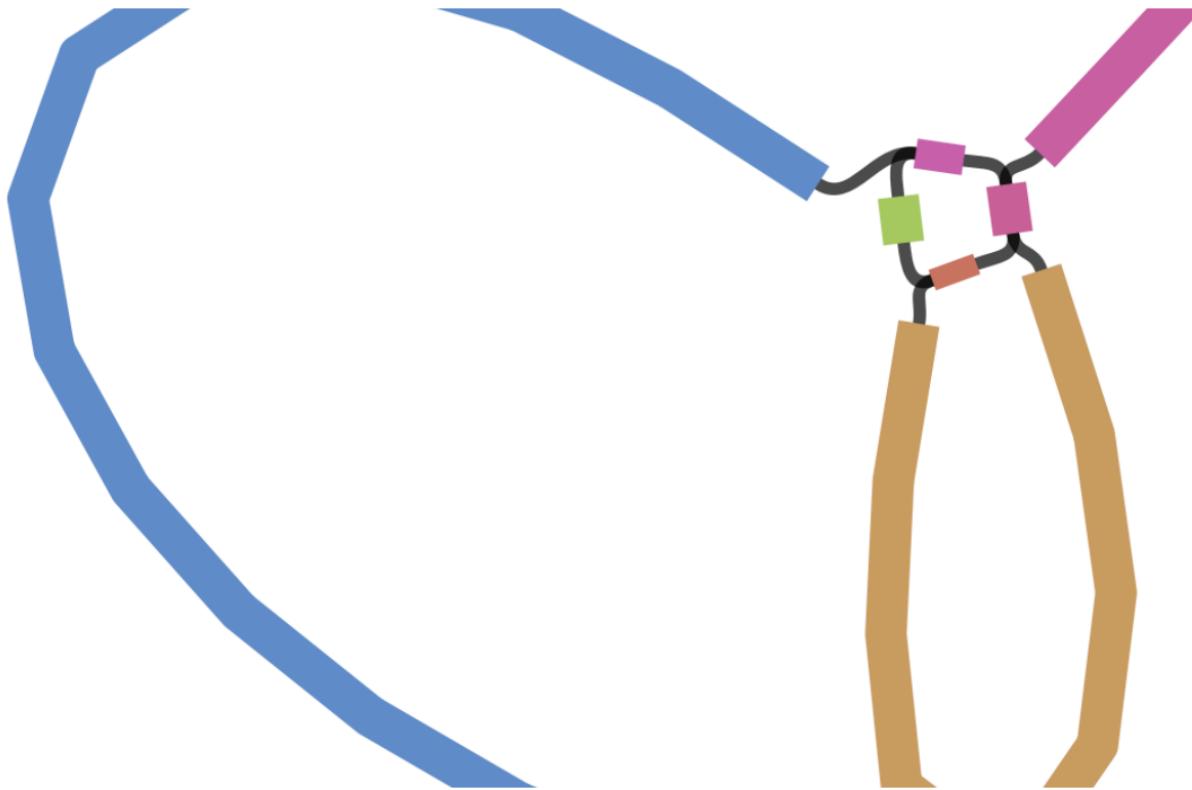
- Read length matters (ii)



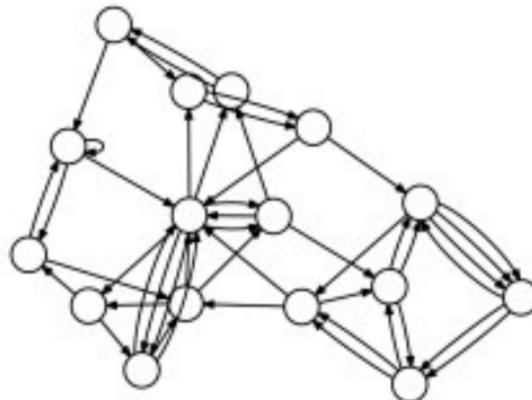
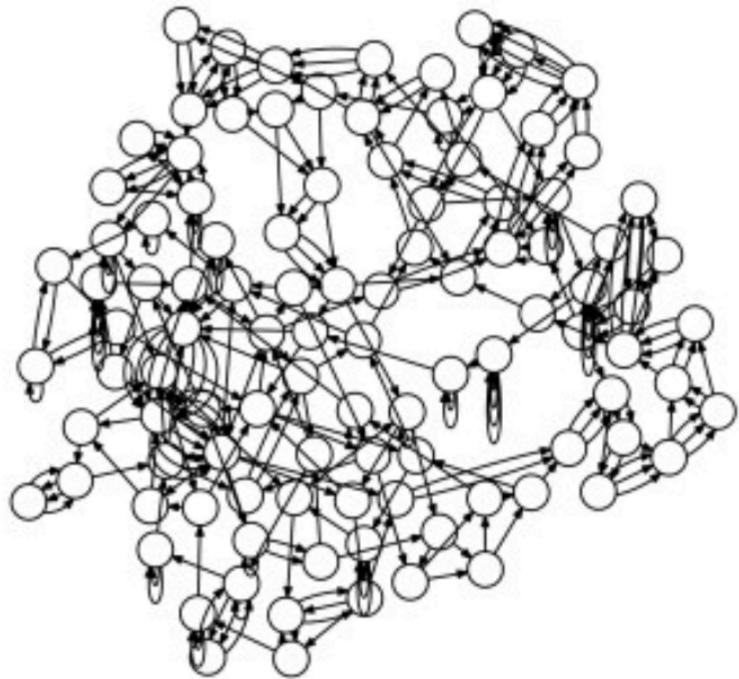
- **Read length matters**

Read size=2000

- Read length matters (ii)



- Great hope for assembly



Golden Threshold



- Great hope for assembly (ii)

From “One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly” Current Opinion in Microbiology 2015

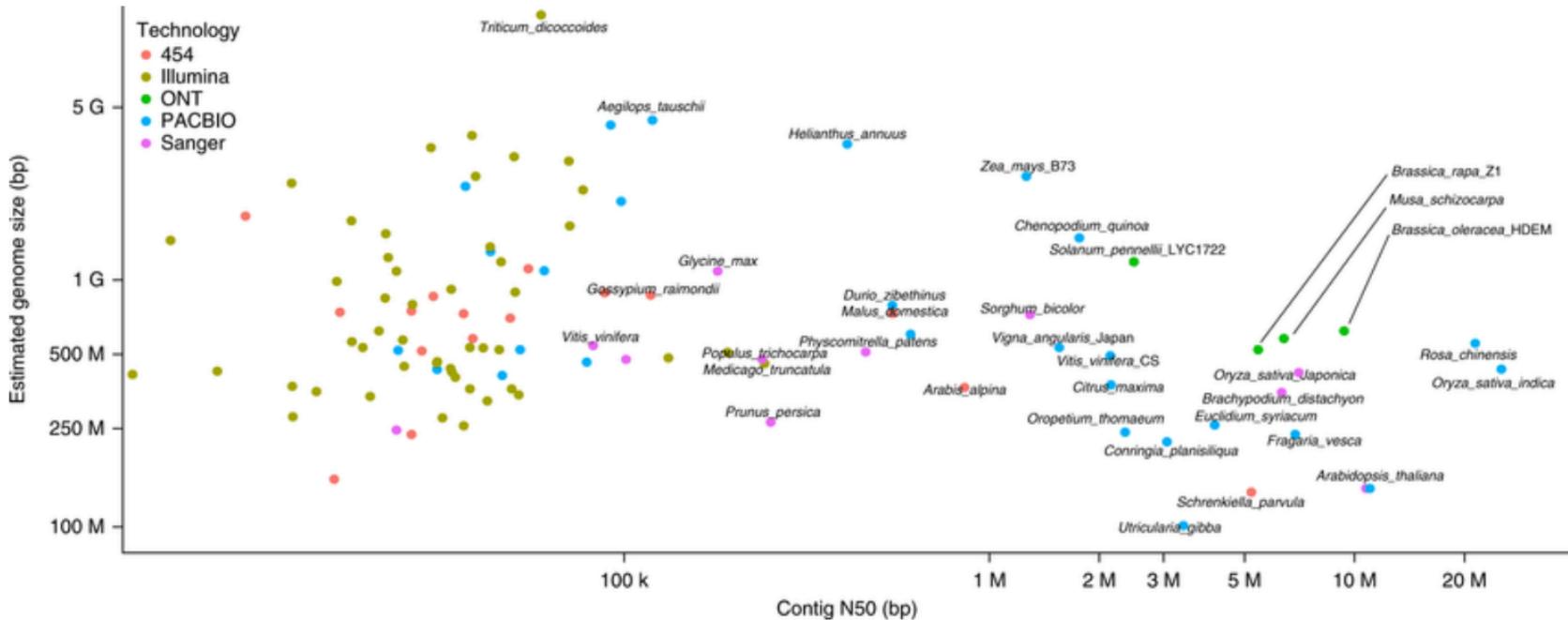
- Long reads killed the assembly star



**Laura Landweber** @LandweberLab · Jan 2

Our newest version of Oxytricha's somatic genome is out ([rdcu.be/bZNfC](https://rdcu.be/bZNfC)) and has 18,617 distinct chromosomes. That's 2000 more than we previously published in [doi.org/10.1371/journal....](https://doi.org/10.1371/journal.pbio.2000001) PacBio captured most chromosomes in single reads: Genome sequence, No assembly required

- Great hope for assembly



From “Chromosome-scale assemblies of plant genomes using nanopore long reads and optical maps” Nature Plants 2018

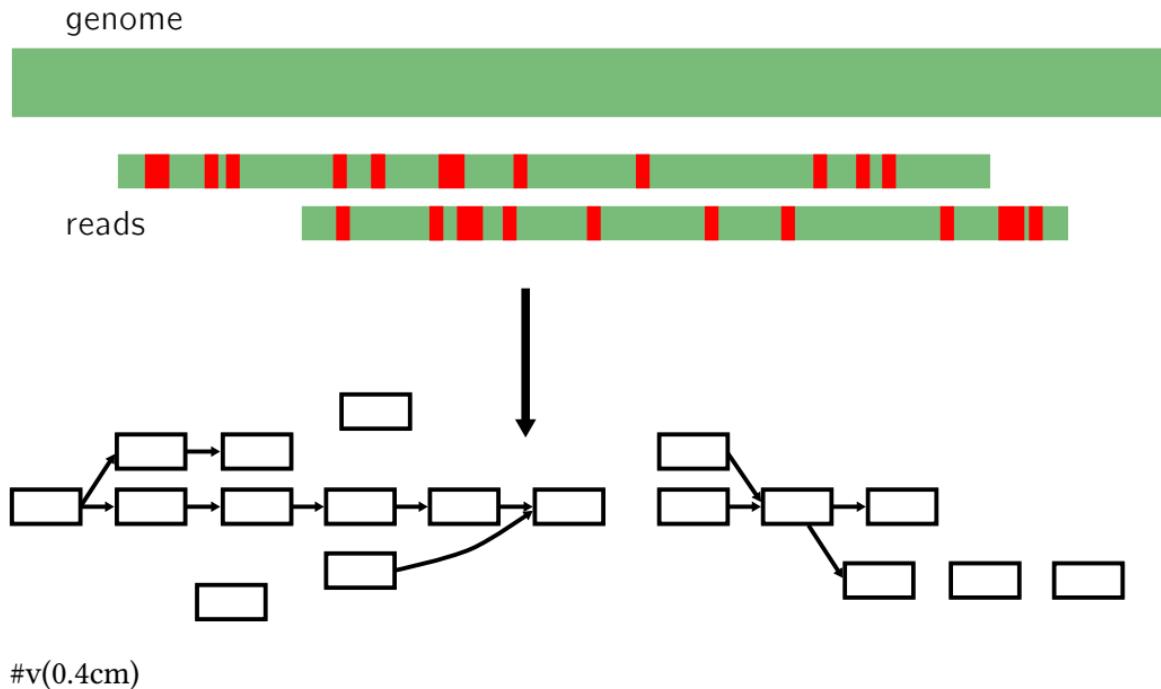
- Which assembly strategy is best suited?

- Long reads approx 10 kbp
- High error rate approx 10 %
- High throughput (up to millions of reads per run) Based on long reads properties, which assembly solution would you choose and why?

Vote!

- Greedy
- Overlap graph
- de Bruijn graph

- Long reads for assembly: de Bruijn graph?



Most  $k$ -mers will contain at least an error and will be useless

- Long reads for assembly: overlap graph?

Supposed to be super expensive!

- Long reads for assembly: overlap graph? (ii)

```

UHHHUCCTGAA
AAAGCTCTGA
AAGCTCTGAA
AGCTCTGAAT
GCTCTGAATC
CTCTGAATCA
TCTGAATCAA
CTGAATCAC
TGAATCAACG
GAATCAACGG
AATCAACGGA
ATCAACGGAC
TCAACGGACT
CAACGGACTG
AACGGACTGC
ACGGACTGCG
CGGACTGCGA
GGACTGCGAC
GACTGCGACA
ACTGGACAA
CTGGACAAAT
TGGACAAATA
...

```

TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTGTCACTTCAGTAAAAACACCTCACGAGTTAAAACACCTAAGTTC  
TAAGAAAGCTCTGAATCAACGGACTGCGAC

GAAAGCTCTGAATCAACGGACTGCGACAAT  
AGCTCTGAATCAACGGACTGCGACAATAAG  
TCTGAATCAACGGACTGCGACAATAAGTGG  
GAATCAACGGACTGCGACAATAAGTGGTGG  
TCAACGGACTGCGACAATAAGTGGTGGTATCCA  
ACGGACTGCGACAATAAGTGGTGGTATCCA

Average coverage: 10

Read length: 10  
Average overlap: 9

Read number: 100

Average coverage: 10

Read length: 30

Average overlap: 27

Read number: 22

- Longer reads, better overlaps

- Less reads for the same coverage
- Larger overlaps

5Mb bacteria example with 100X coverage

**Short reads**

body

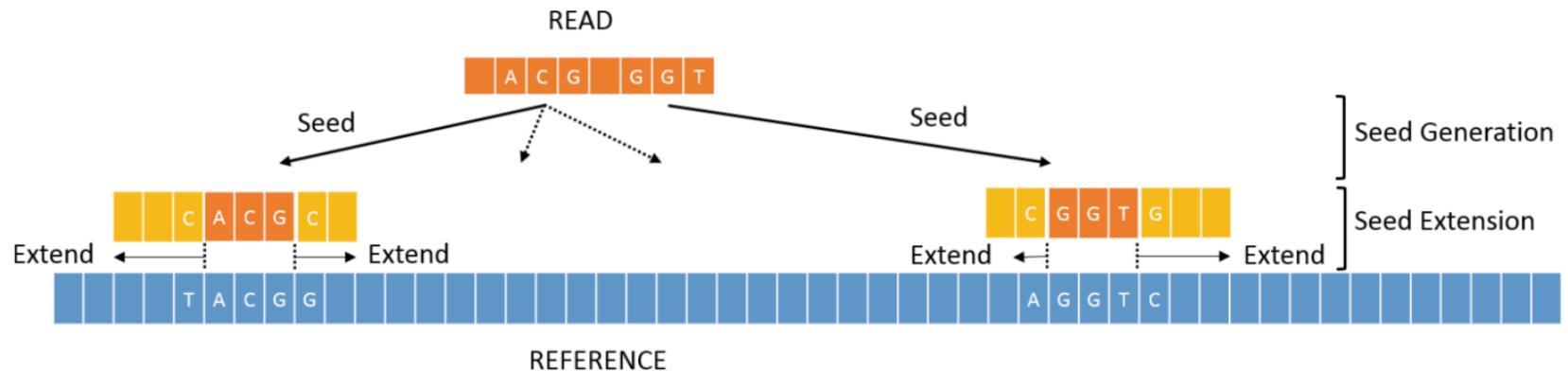
**Long reads**

body

**Very long reads**

body

- Are large overlaps hard to compute?



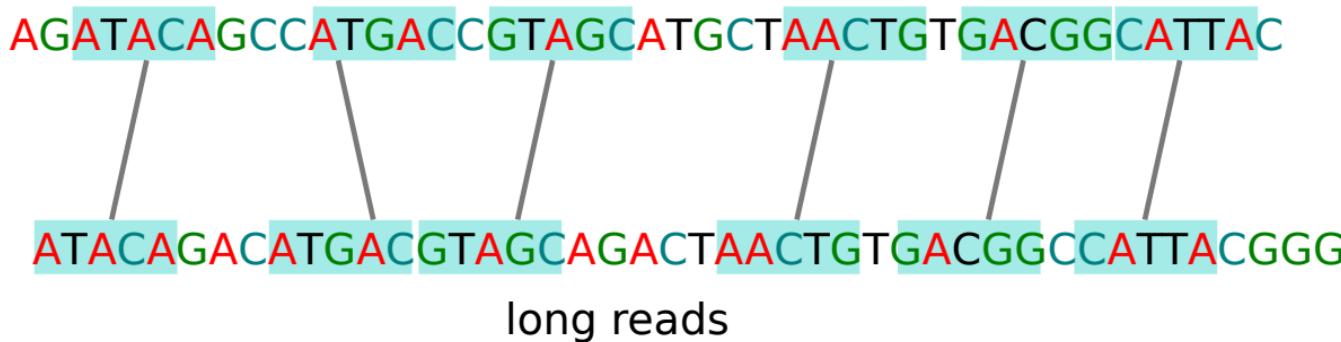
Aligning very long and highly erroneous regions is expected to be expensive, as alignment is quadratic approx  $O(n^2)$  !

- “Anchor chaining” in overlap graph

GCCATGACC

short reads

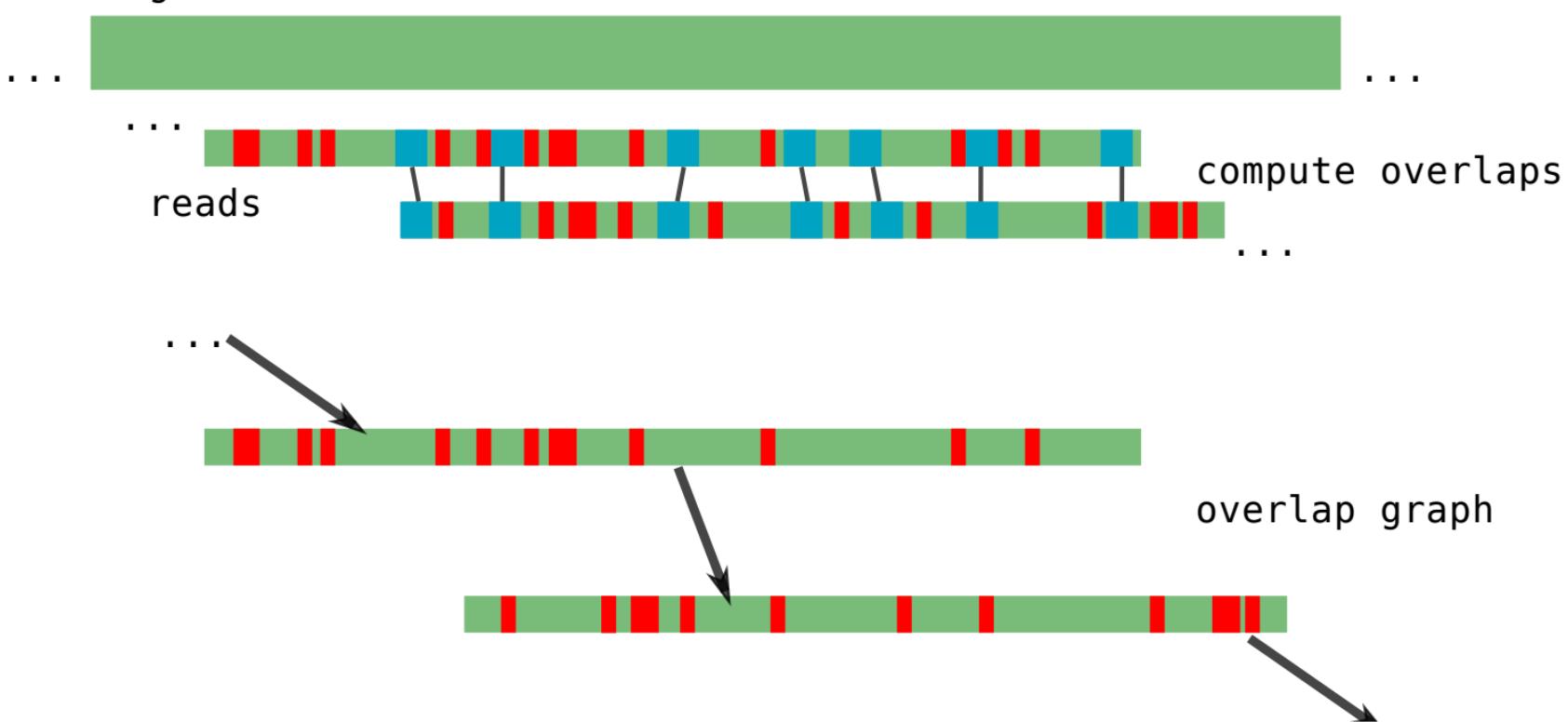
CATGACCGA



- For long reads: typically Minimap2's [Li 2018] job

- “Anchor chaining” in overlap graph (ii)
- “Anchor chaining”: find common chains of anchors ( $k$ -mers) in the same order in 2 sequences (can be **linear** in practice in most cases)

- Long reads for assembly: overlap graphs



- Sequencing errors

Genome:

ATCGGTATCGTTACGGTATAACC

Reads:

ATCGCTATCG  
GGTATCGTCTA  
ATGTTACGG

(Substitution)  
(Insertion)  
(Deletion)

- **Sequencing errors (ii)**

High rate of insertions and deletions rendered genome annotation nearly impossible

- Using coverage to remove noise: Consensus

Reads:

AAAGAAAGCACTGAATCA|TGGGACTT|CGAG  
 GAAAGCTCTCAACCAACGGACTGCGACTTT  
 ACCTCTCAAGCAACGGACTGCGACAAAAAG  
 TCTGAATCACCGACTGCGTCAAAAAGTGC  
 GAATCACCGACTGCGACAGTTGTGGTGG  
 TCAACCGCACTGCGACAATAAGTCCTGGTAT  
 ACGGACTGCGACAAAAAGTGTGGTATCCA  
 GACTGCCACAAAAAGTGGTGGTATCCAG  
 TGCGACAAAAAGTGGGTATCCAGAAT  
 GACAATAAGGGGGGTATCCAAAATTG  
 AAAAAGGGGTGGTATCCAGAATTTCAG  
 TAAGTGGGGGTATCCAAAATTTCAGTT

Consensus:

AAAGATAGCTCTGAATCAACGGACTGCGACAAAAAGTGGTGGTATCCAGAATTTCAGTT  
 1/1            4/7            9/10            6/11            3/4

- Exercise 2: Perform a consensus

RS1: ACTTCGAACCGT

RS2: TCGATCGTTT

RS3: GATCAGTTTAG

RS4: TCATTTCGTA

RS5: GTTTCGTCCG

REF: ACTCGAACATGTTTCCCTACG

- Exercise 2: Perform a consensus - solution

RS1: ACTTCGA-AC-GT-----

RS2: --T-CGA-TC-GTTT-----

RS3: -----GA-TCAGTTT-AG---

RS4: -----TC-ATTT-CGTA--

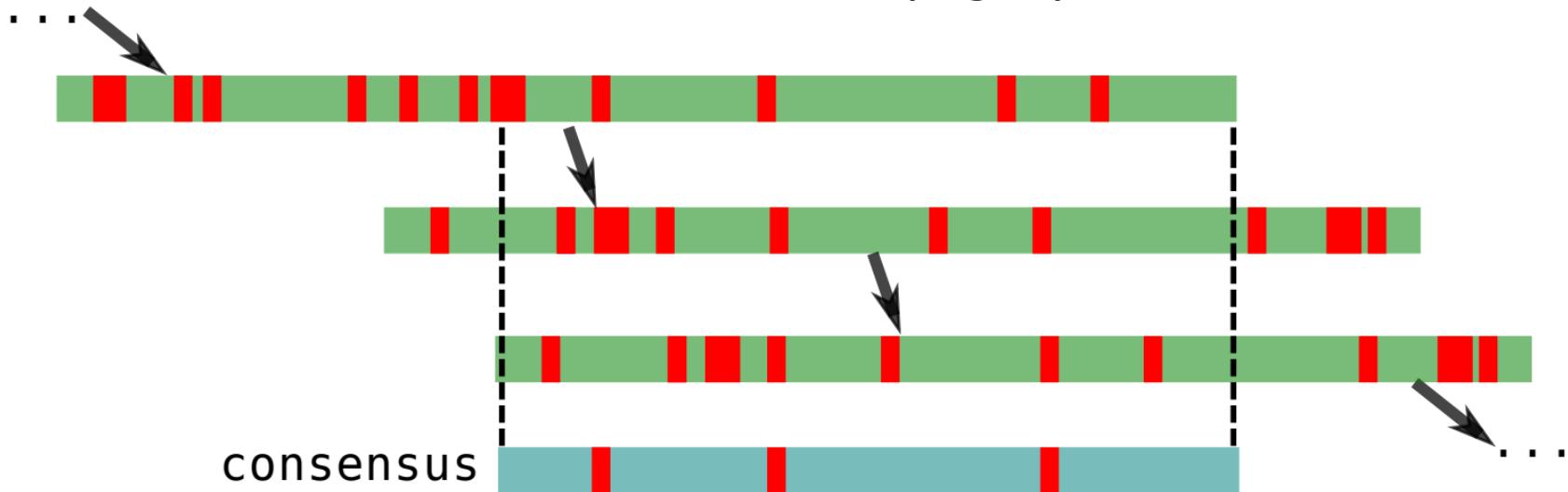
RS5: -----GTAA-CGTCCG

REF: ACT-CGAAT--GTTTTCTACG

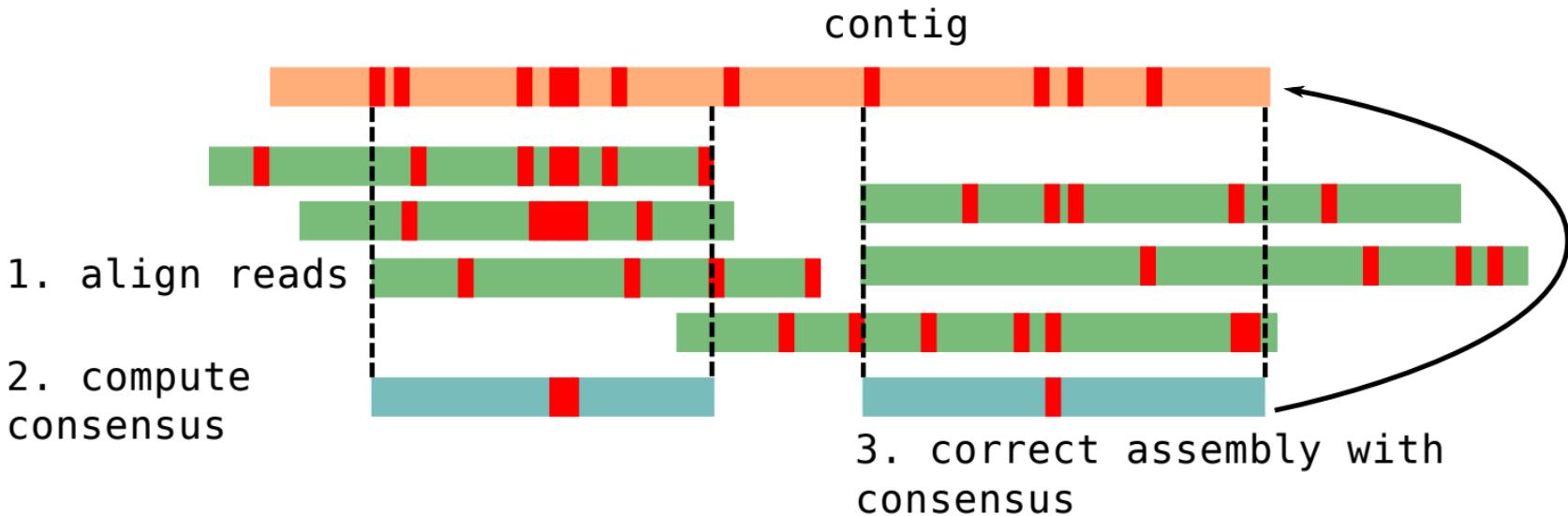
CON: ACT-CGAATC-GTTT-CGTACG

- Consensus during assembly

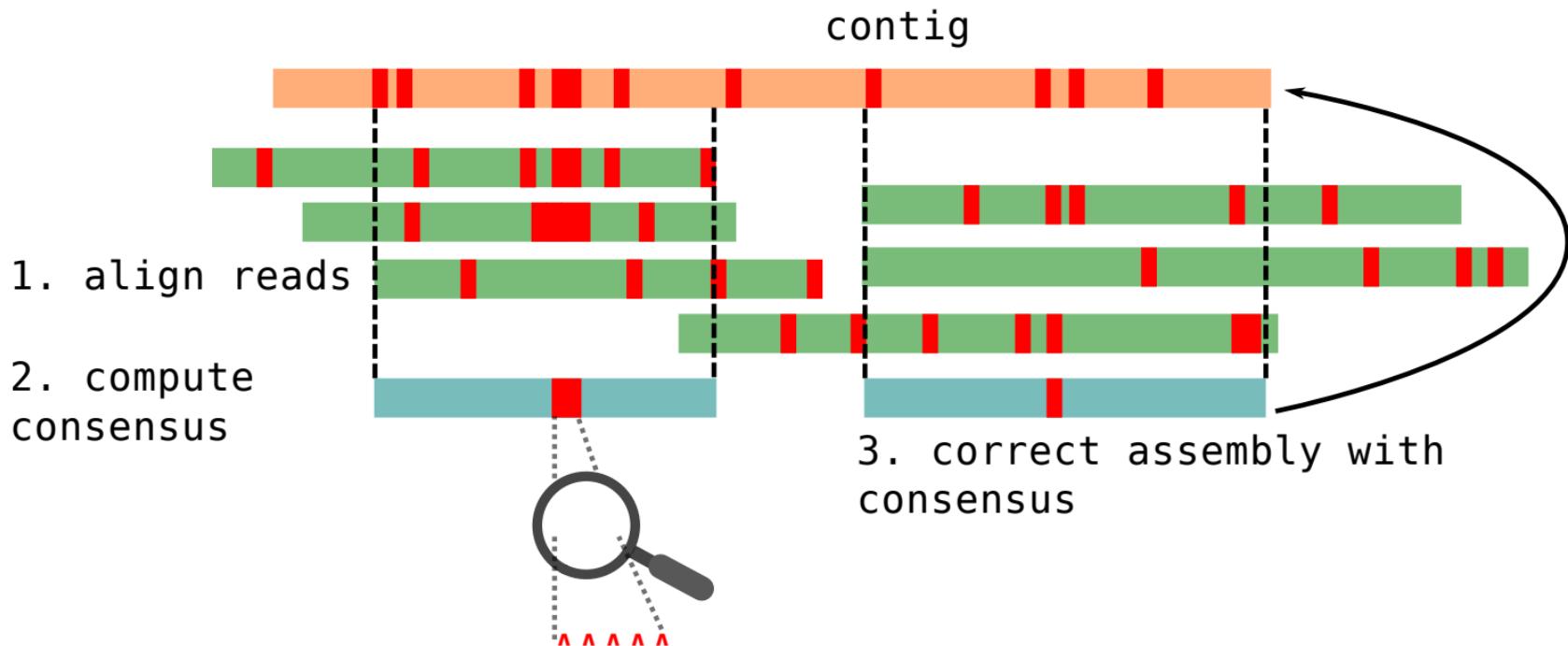
overlap graph



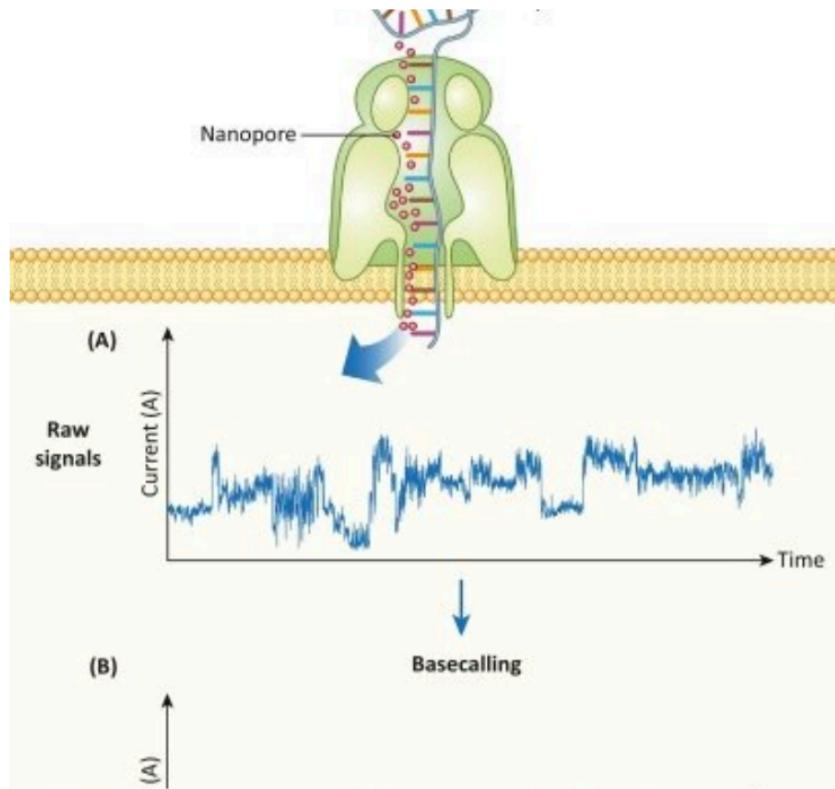
- Consensus after assembly: polishing



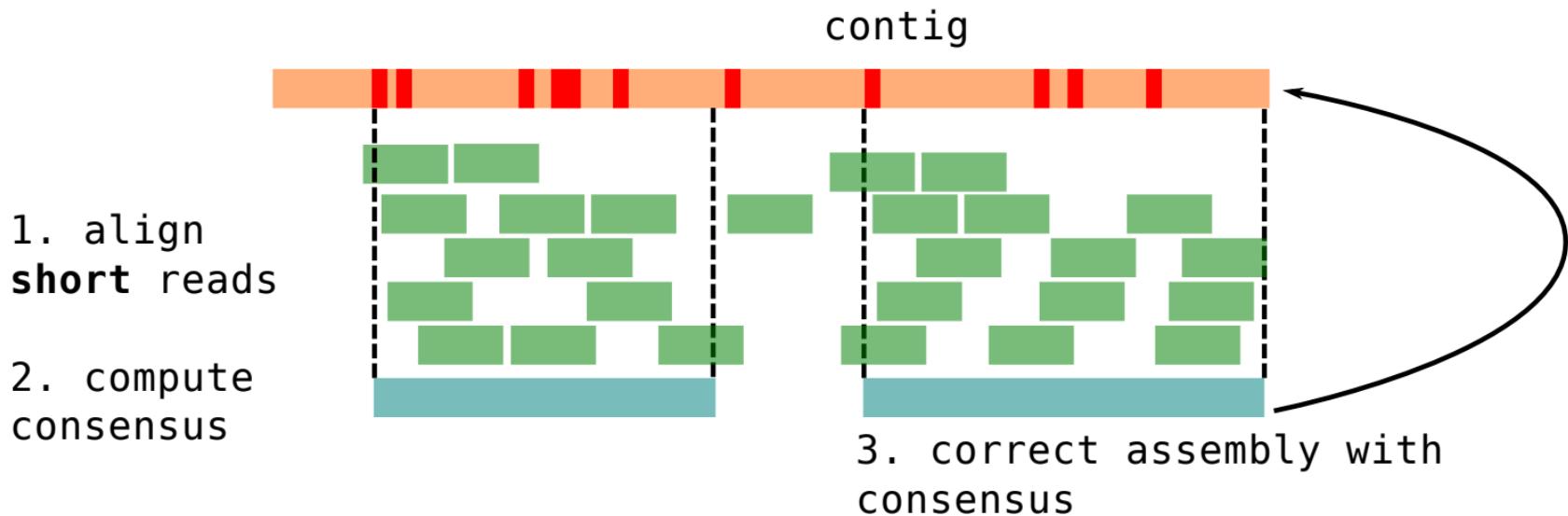
- Consensus after assembly: polishing



- Homopolymers are hard to read



- Polishing using accurate reads



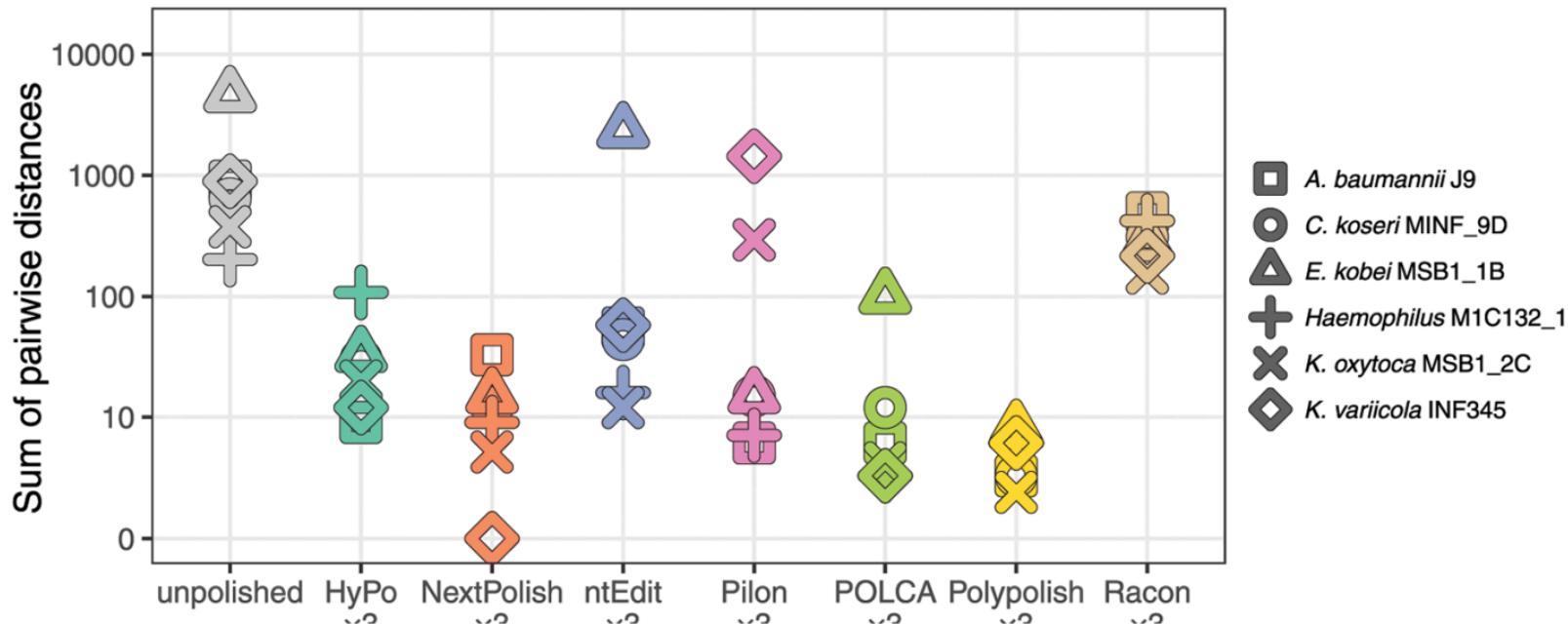
- **Systematics errors**

Polishing with Illumina data can improve the final error rate

## • Systematics errors (ii)

### A. Single-tool short-read polishing

ALE change: 0 110696 113366 87707 113056 113061 115623 82446  
 total distance: 7635 212 74 2519 1775 128 28 1867



- **Systematics errors (iii)**

From Polypolish: Short-read polishing of long-read bacterial genome assemblies

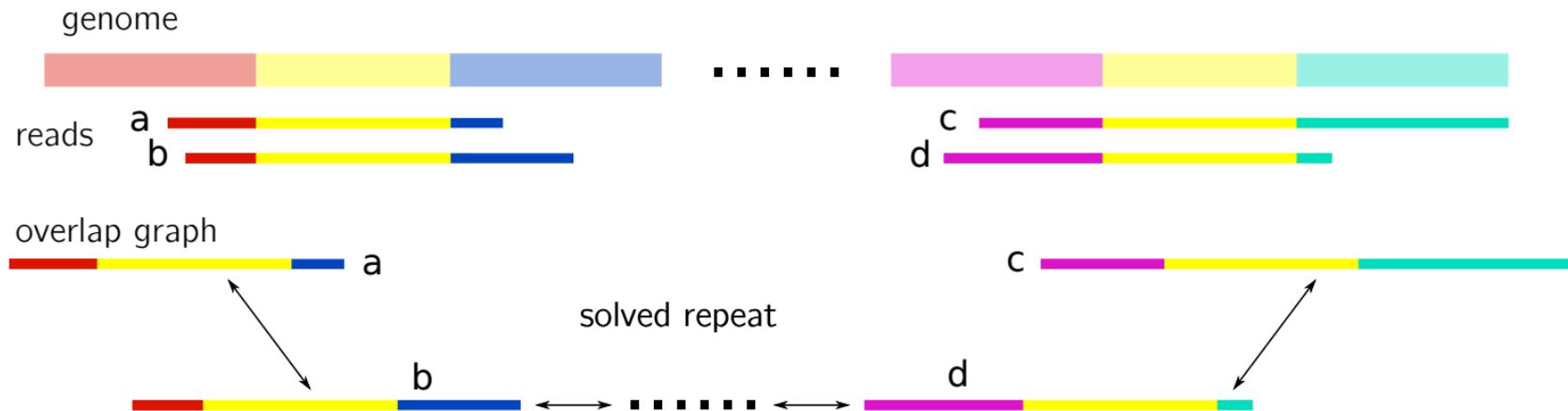
- I know we said it was the end

Just one or two more graphs

- I know we said it was the end (ii)



- An overlap graph limitation. Swept under the carpet?



- An overlap graph limitation. Swept under the carpet?

genome

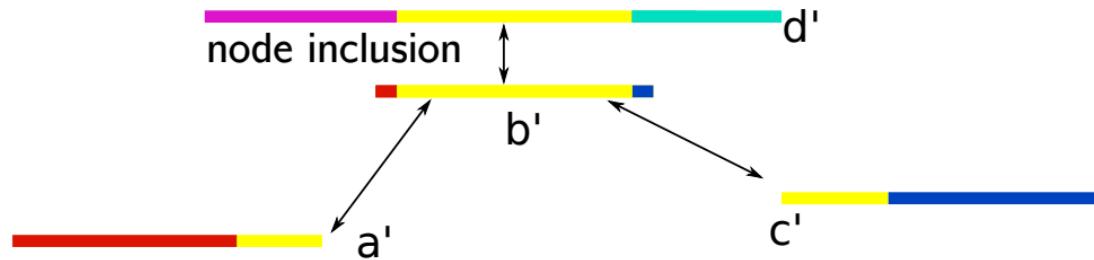


reads

a' c'

d'

b'



- An overlap graph limitation. Swept under the carpet?

genome



a'

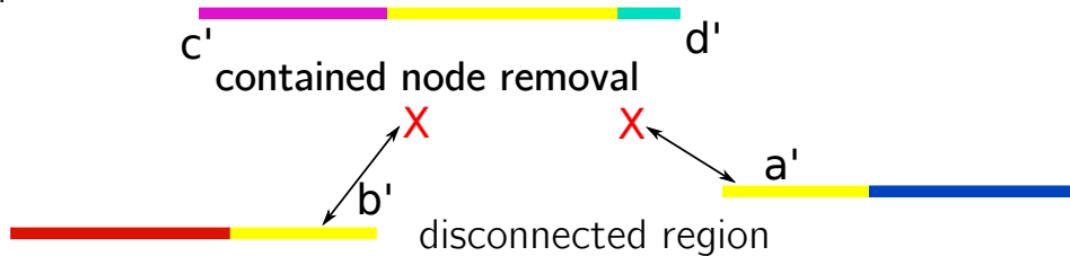
c'

d'

reads



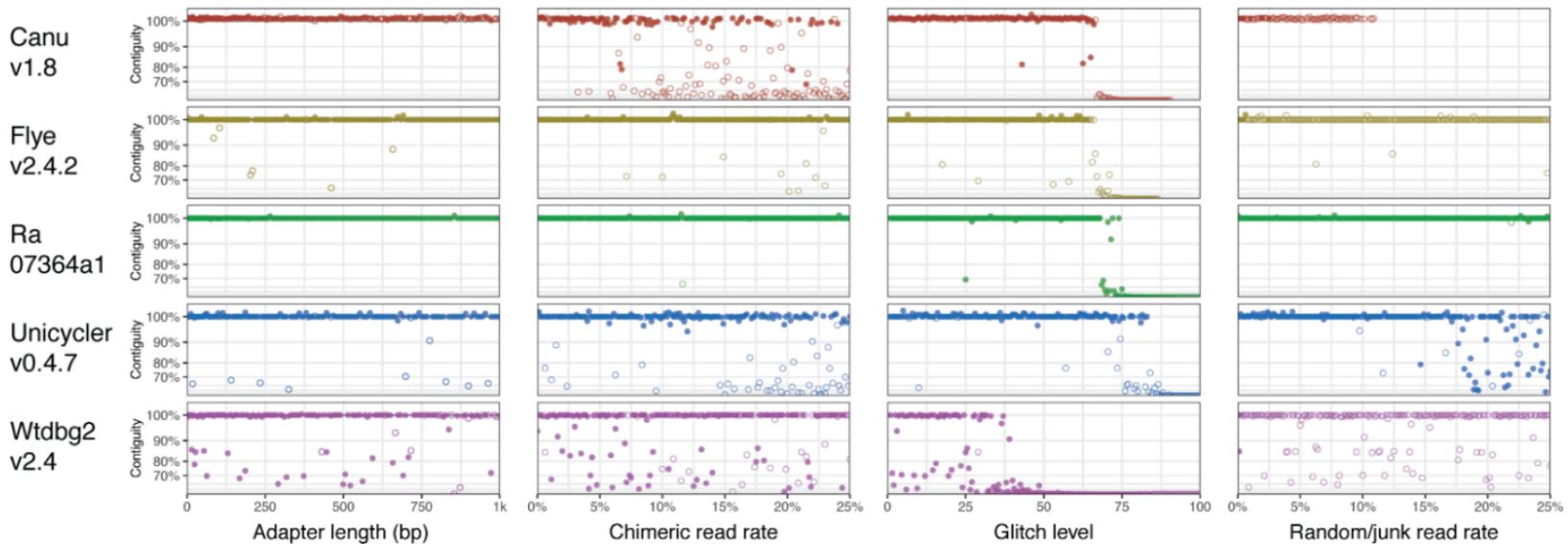
overlap graph



- Long reads for assembly: assembly solved?

**Assembly is not solved yet**

Sometimes the software fails



From <https://github.com/rrwick/Long-read-assembler-comparison>

- Long reads for assembly: assembly solved?

**Assembly is not solved yet**

Sometimes the data cannot solve the problem

- Very large repeated region
- Low local coverage
- Chimeric/noisy reads

- 20 years later



- 20 years later (ii)



- **Telomere-to-Telomere consortium**

**Has produced in 2021 a complete human genome with one contig per chromosomes !**

- 30x PacBio HiFi
- 120x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 10X Genomics, BioNano DLS and Arima Genomics HiC
- 100 authors from 50 labs

- Long reads assemblers

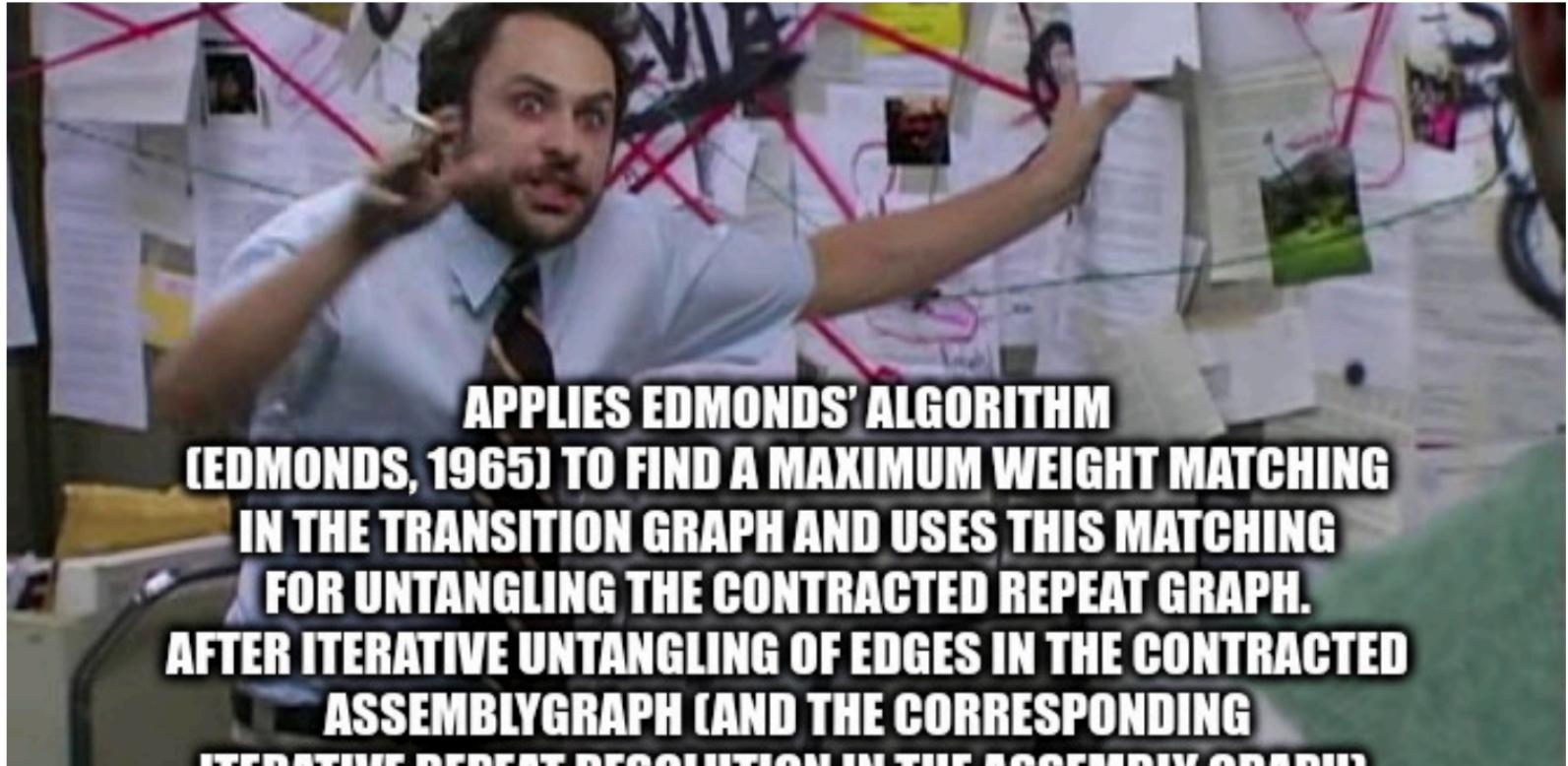
**Best performing assemblers**

body

**Other notable assemblers**

body

- Flye



- Repeat graph

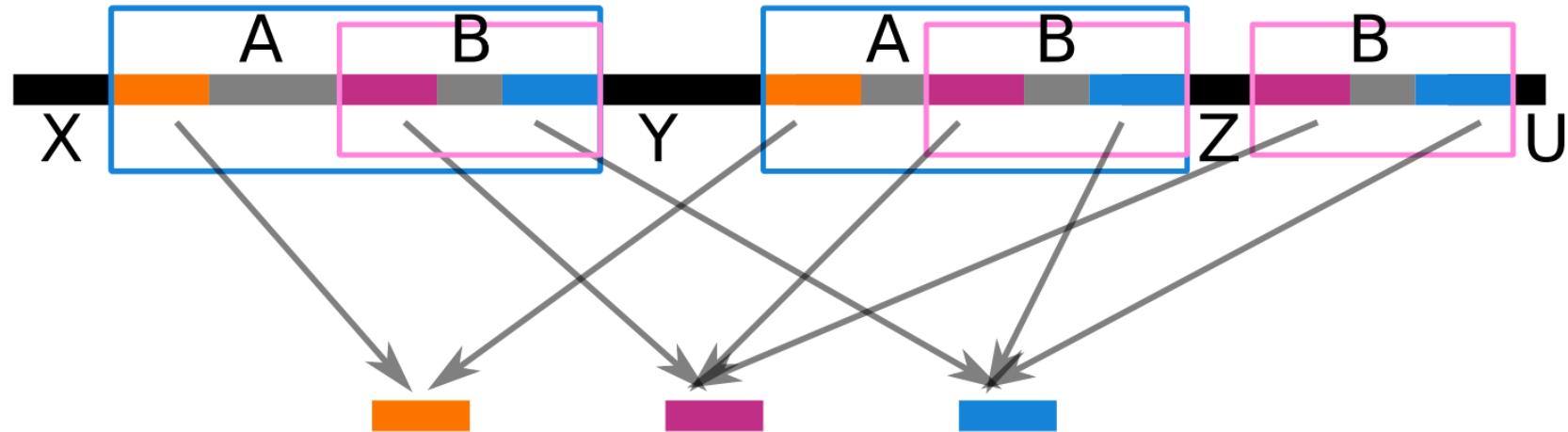
a genome



highlighted repeated regions

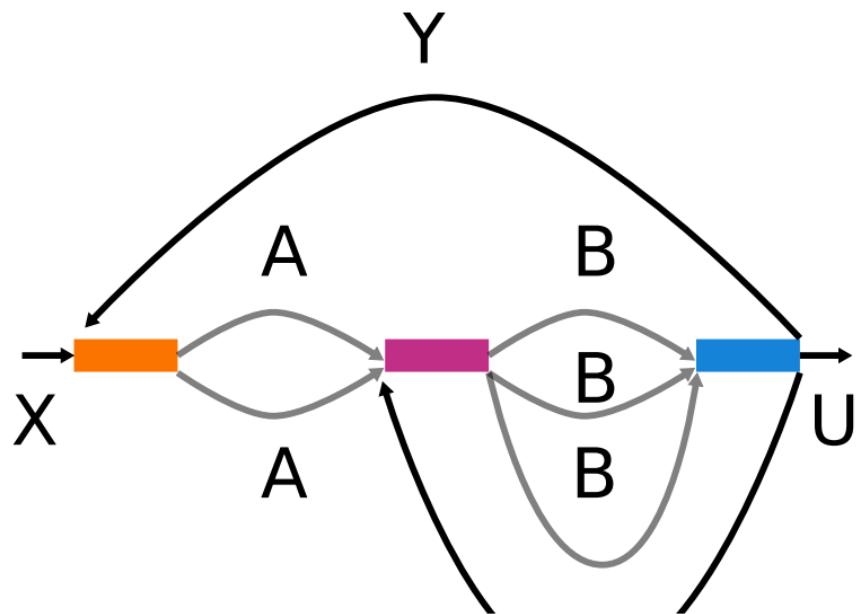


- Repeat graph

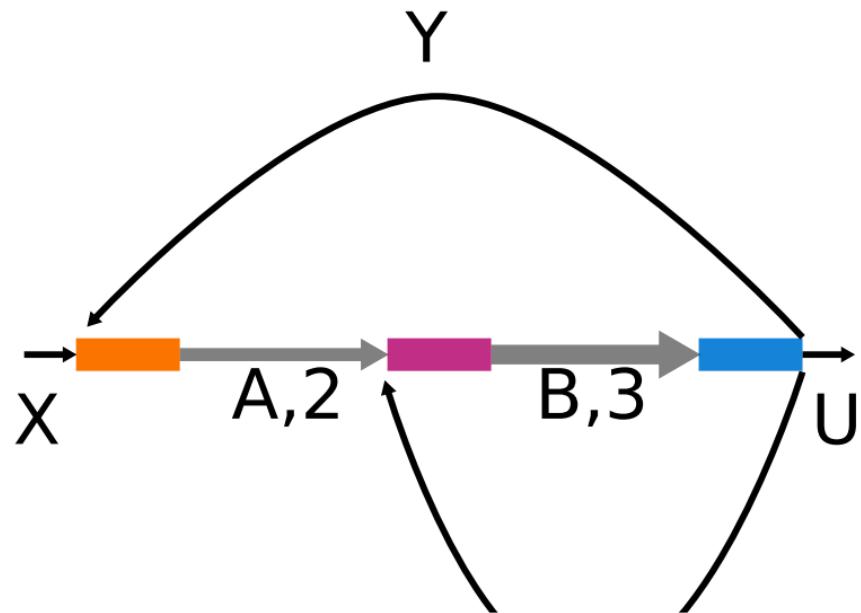


repeats extremities: graph's nodes

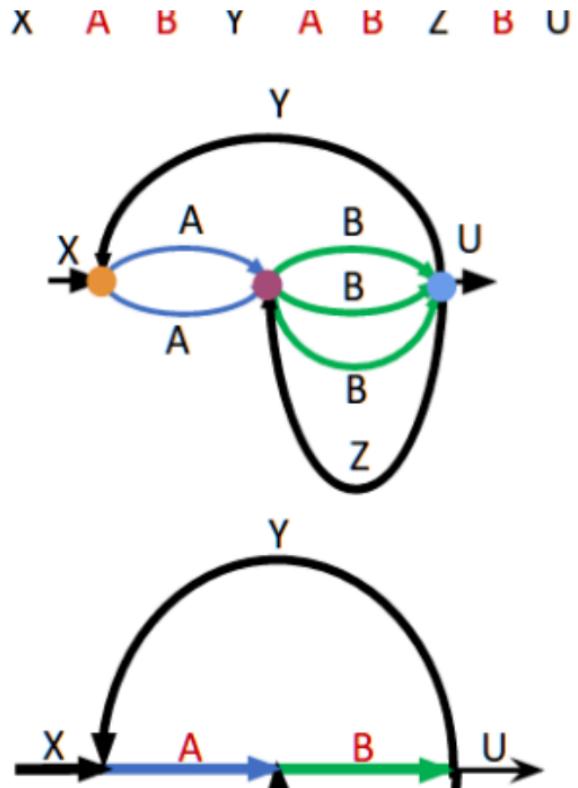
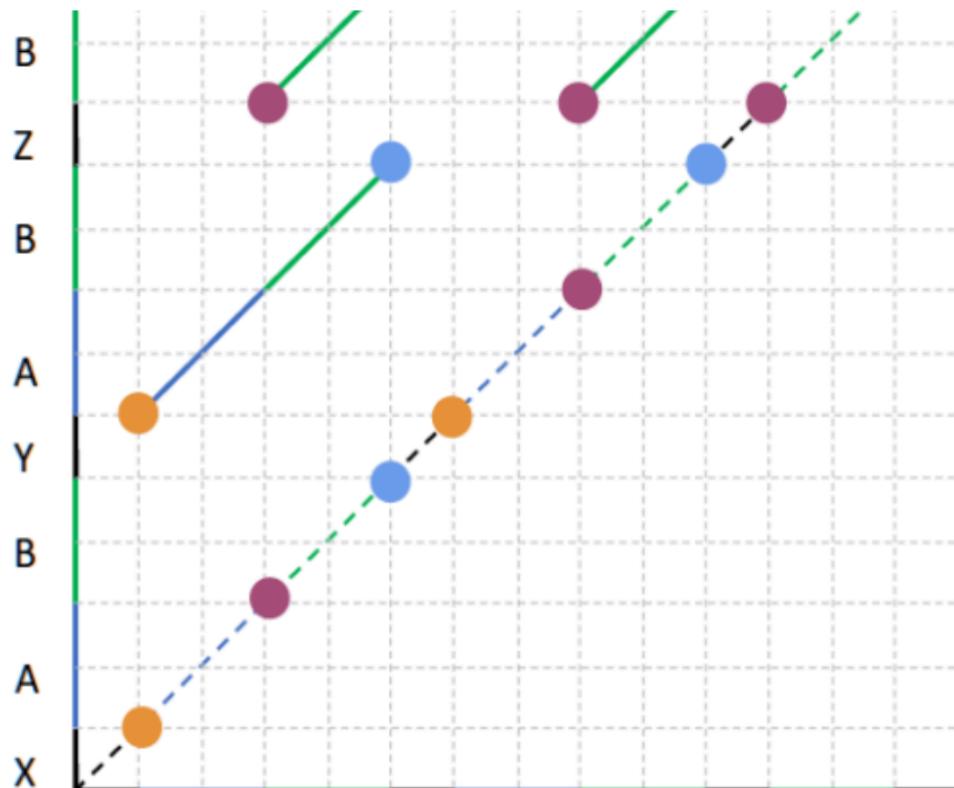
- Repeat graph



- Repeat graph



- Repeat graph

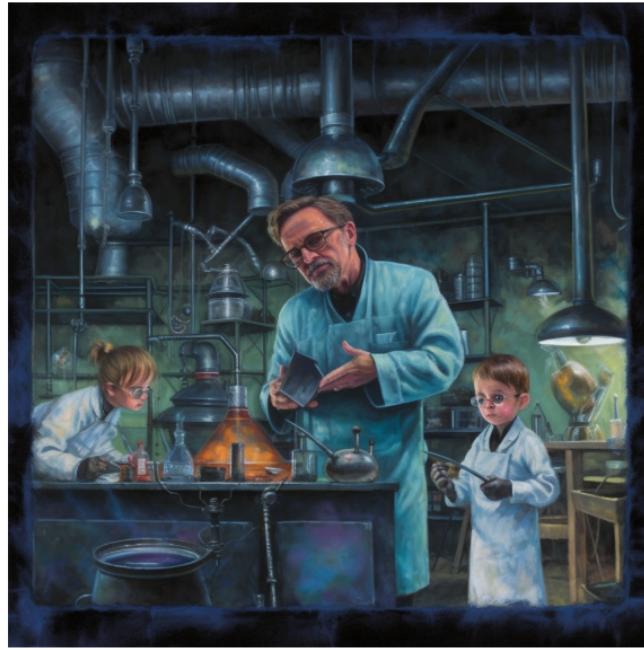


- **Long read assembly summary**

- Overlap graphs with quick overlap computation
- Long reads can span repeats and improve assemblies
- Methods to polish contigs

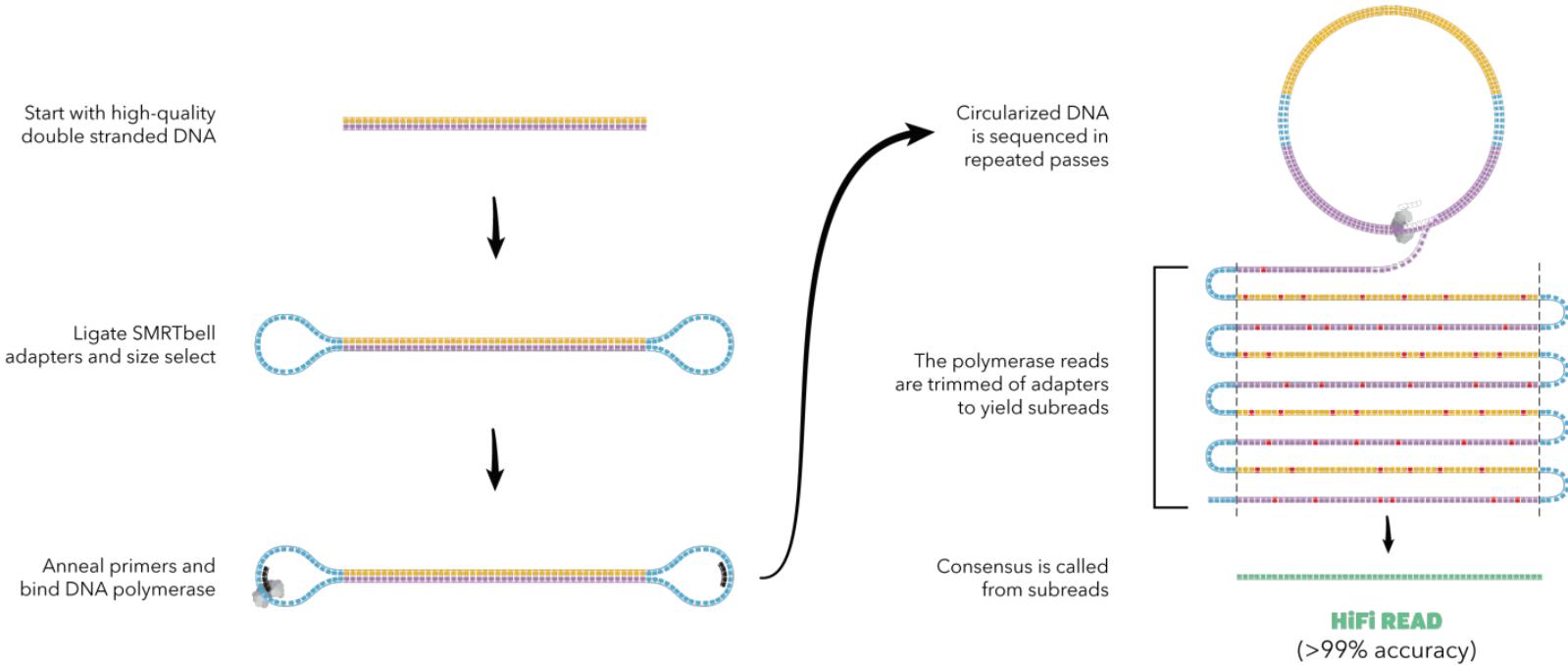


- Future of assembly



The future of genome sequencing according to MidJourney. Left: yes, but kinda boring. Right: hmmm.

## • Consensus during sequencing



- Consensus during sequencing (ii)

HiFi data

body

## • HiFi Assembly

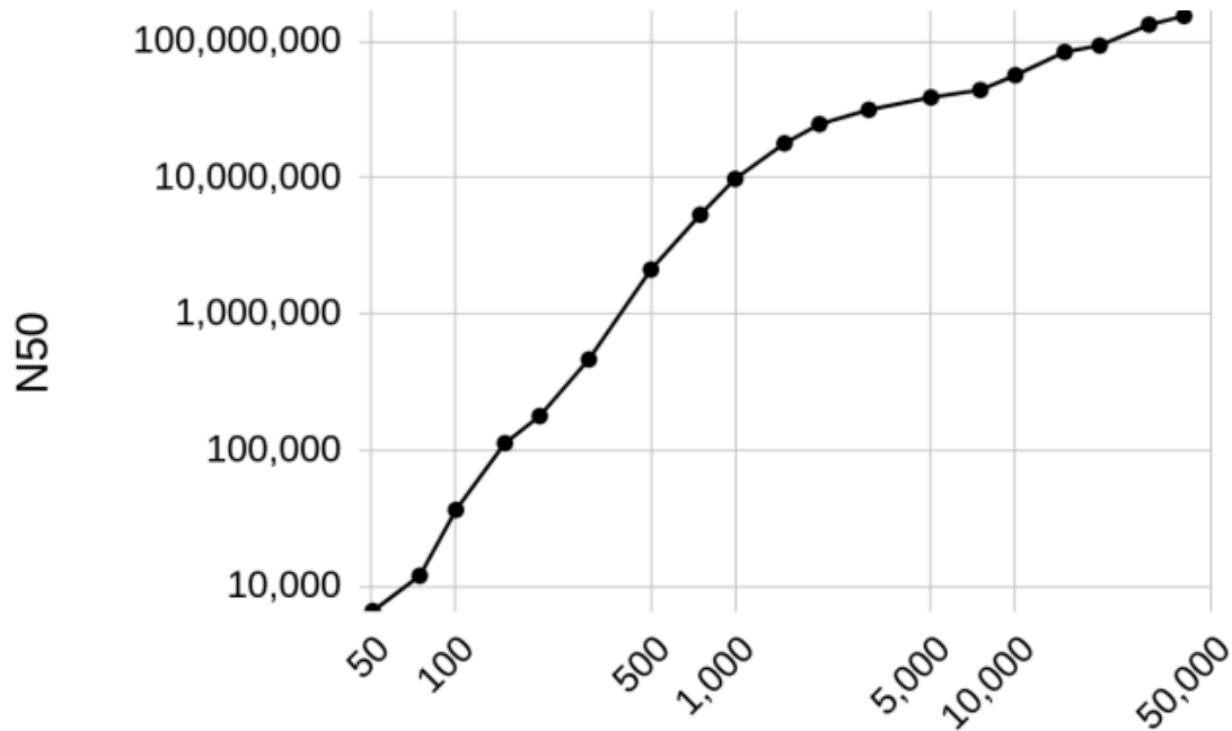
With almost error-less long reads we have several promising improvements ahead:

- Use de Bruijn graph (more efficient data structures)
- Assemble large genomes very fast
- Perform diploid assembly

- de Bruijn graph Assembly

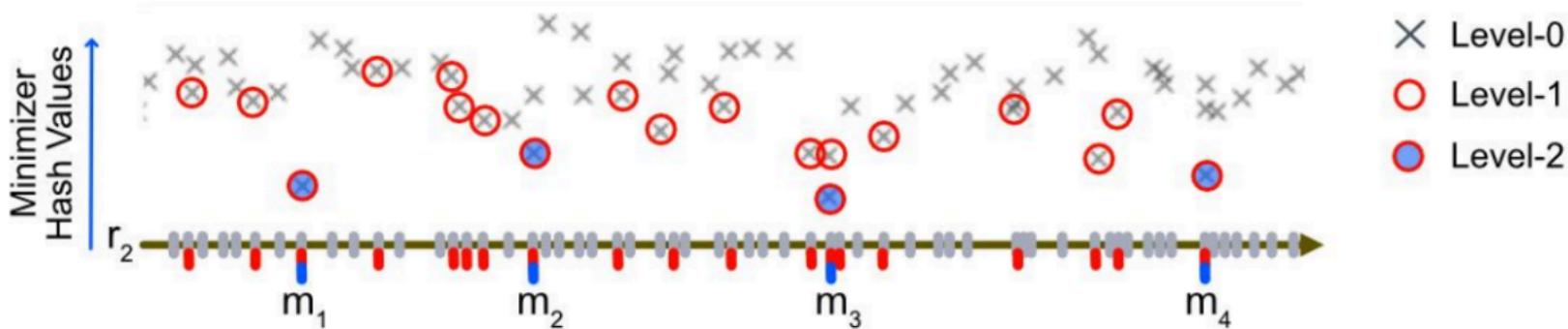
Using K=500 and K=5000 de Bruijn graphs to assemble

- de Bruijn graph Assembly (ii)



- Very fast genome assembly

Human genome assembled within 2 hours (Peregrine assembler) and 10 minutes (RMBG assembler)



- Diploid assembly

(B)

Diploid outbred genome



Unphased maternal and paternal contigs



Haplotype-mosaic reference assembly



Assembled maternal and paternal chromosomes



- Human diploid assembly (20/46 chr)

Asm	Contig NG50 (Mb)	Scaffold NG50 (Mb)	Hamming error	QV	#Errors Chr X & Y
<b>Downsampled (35x HiFi / 60x ONT-UL)</b>					
Verkko	12.90		0.13%	52.18	10
Verkko + trio	<b>80.77</b>	<b>102.55</b>	0.13%	52.40	10
Verkko + Hi-C	58.24	82.42	0.16%	52.48	10
LJA	0.39		0.14%	55.75	7
Hifiasm (unitigs)	0.35		0.23%	<b>61.05</b>	<b>2</b>
Hifiasm + trio	64.50		<b>0.06%</b>	60.86	26
Hifiasm + Hi-C	66.34		0.79%	60.57	37
<b>Full-coverage (&gt;100x HiFi / 85x ONT-UL)</b>					

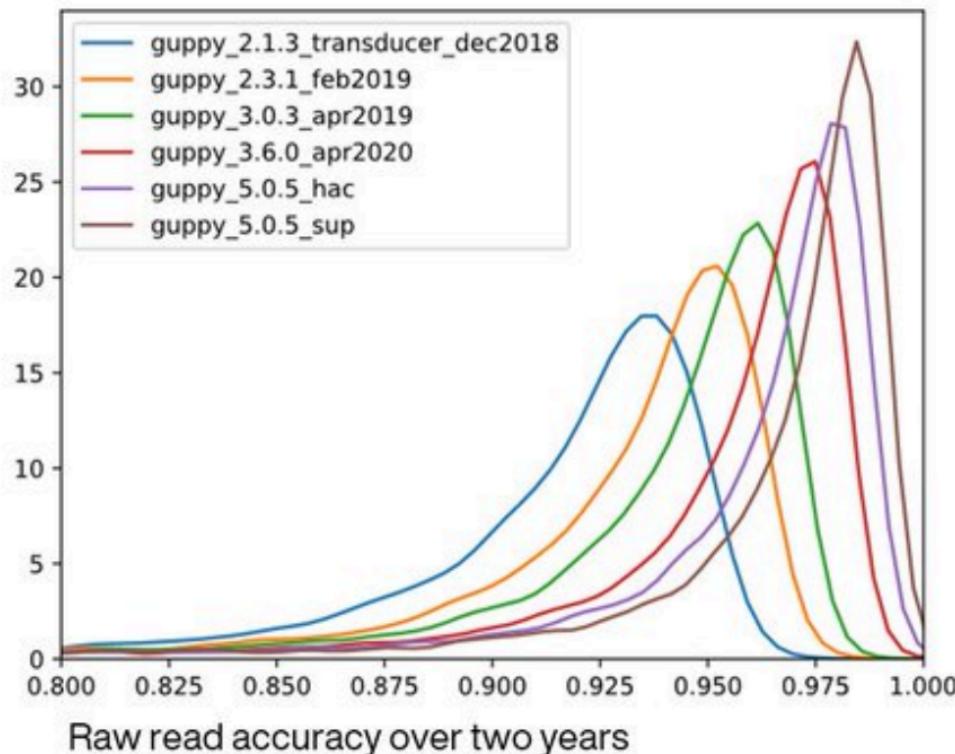
- Human diploid assembly (20/46 chr) (ii)

Verkko	17.35	0.05%	54.91	5
Verkko + trio	<b>134.00</b>	135.80	0.05%	55.77
Verkko + Hi-C	68.32	85.97	0.05%	55.57
HPRC curated	72.70	<b>146.75</b>	0.13%	61.35

- Ongoing progress

Errors in Nanopore sequencing data are rapidly diminishing

- Ongoing progress (ii)

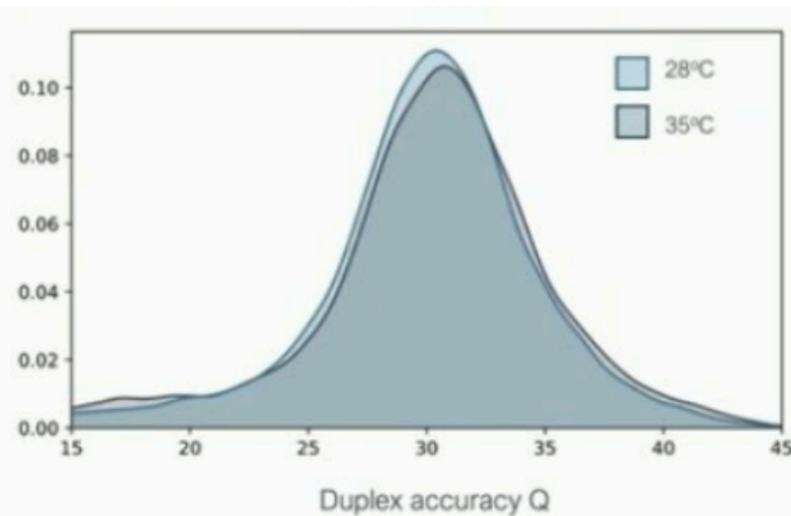
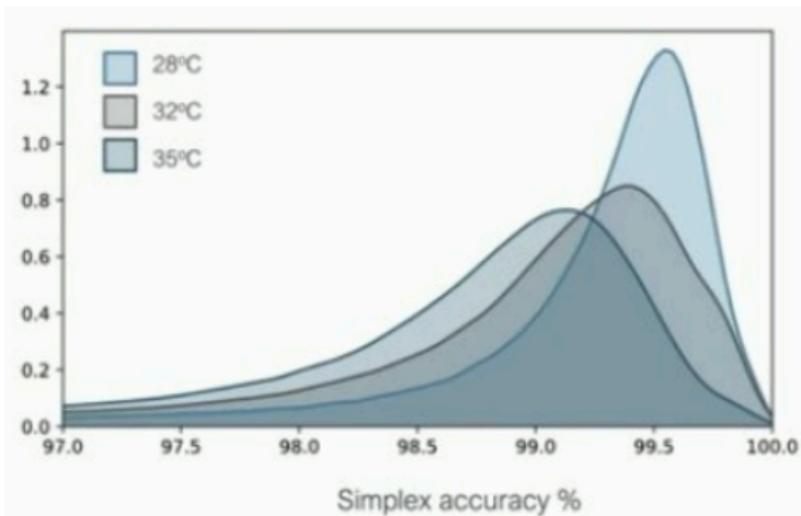


- Ongoing progress (iii)

Q20 chemistry achieved modal accuracy approx 99%

- High fidelity Nanopore incoming?

Nanopore duplex reads could deliver long and precise reads in the future



- Take home messages

Ultra fast summary

body

- Ongoing work

### Assembly Challenges

body

- The end



PopGenGoogling  
@popgengoogling



i trust you to figure out your own genome

[Traduire le Tweet](#)

3:01 AM · 14 déc. 2019 · Twitter for iPhone

---

**37** Retweets    **267** J'aime

---